conference

proceedings

The First USENIX Workshop on Electronic Commerce

New York, New York
July 11–12, 1995

USENIX®
The UNIX® and Advanced Computing Systems Professional and Technical Association

For additional copies of these proceedings write:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 USA
510 528 8649
office@usenix.org

The price is $20 for members and $26 for nonmembers.
Outside the U.S.A. and Canada, please add
$10 per copy for postage (via air printed matter).

ISBN 1-880446-74-X

Printed in the United States of America on 50% recycled paper, 10-15% post consumer waste.

USENIX Association

Proceedings of the First USENIX Workshop of Electronic Commerce

July 11-12, 1995

New York, New York, USA

First USENIX Workshop on Electronic Commerce
July 11-12, 1995

Table of Contents

Wednesday, July 12, 1995

Technology
Session Chair: Peter Honeyman

Extensions
Session Chair: Marc Donner

PREFACE

Voltaire, in saying "There is nothing so powerful as an idea whose time has come," might well have been presaging electronic commerce. The question is whether there are technologically immovable objects in the path of this irresistible force, and whether there are caveats to apply that only a technically sophisticated audience would see.

This, the First USENIX Workshop on Electronic Commerce, should provide a major opportunity for researchers, experimenters, and proto-practitioners in this rapidly self-defining field to exchange ideas and present results of their work. That is certainly our aim and our hope.

More importantly, and why I want to welcome each and every one of you to this event, is that while "electronic commerce" is an idea of Voltaire quality, and the *New York Times* is using headlines like "The Serious Money Hits the Superhighway," there is still time to try to get the engineering right before the calculus of cumulative investment takes over. You, however many of you there are here or wanting to be here, are the only ones who understand that fact.

Let's go; it is both a privilege and a responsibility to be present at the creation.

Thanks to all who made this happen. And I am especially grateful to Peter Honeyman for conceiving, organizing, and delivering both the closing session and enclosed Digest.

Dan Geer
Program Chair

ACKNOWLEDGMENTS

PROGRAM CHAIR
Daniel E. Geer, Jr. ScD., *Open Market, Inc.*

WORKSHOP ORGANIZING COMMITTEE
Joseph Arceneaux, *Wells Fargo Bank*
Nathaniel Borenstein, *First Virtual Holdings, Inc.*
Marc D. Donner, *Morgan Stanley & Co., Inc.*
Steve Dusse, *RSA Data Security, Inc.*
Ittai Hershman, *Advanced Networks & Services Inc.*
Peter Honeyman, *University of Michigan, CITI*
Alan Nemeth, *Digital Equipment Corporation*
Clifford Neuman, *Information Sciences Institute*
Greg Rose, *Sterling Software Pty. Ltd.*
Allan M. Schiffman, *Enterprise Integration Technologies Corp.*
Mark Seiden, *Seiden and Associates, Inc.*
Peter Stalker, *E.M. Warburg, Pincus & Co.*
Win Treese, *Open Market, Inc.*

TUTORIAL COORDINATOR
Daniel V. Klein, *USENIX Association*

PROCEEDINGS PRODUCTION
Carolyn S. Carr, *USENIX Association*
Data Reproduction

USENIX MEETING PLANNER
Judith F. Desharnais, *USENIX Association*

USENIX EXECUTIVE DIRECTOR
Ellie Young, *USENIX Association*

USENIX Marketing Director
Zanna Knight, *USENIX Association*

USENIX SUPPORT STAFF
Colleen Biddle, *USENIX Association*
Eileen Curtis, *USENIX Association*
Diane DeMartini, *USENIX Association*
Toni Veglia, *USENIX Association*

Token and Notational Money in Electronic Commerce

L. Jean Camp
lc2m@andrew.cmu.edu

Marvin Sirbu
sirbu@andrew.cmu.edu

J. D. Tygar
tygar@andrew.cmu.edu

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

What properties of money are important for electronic commerce? We argue that both transactional and privacy properties distinguish electronic commerce systems. We provide a quick overview of the history of money. We then consider privacy provided by different forms of money, and socially desirable disclosure of information as specified by legal reporting requirements. We classify electronic and traditional commerce systems into two categories:

- *token systems, which exchange markers representing value*
- *notational systems, where value is stored as notations in a ledger or computer.*

We analyze different forms of traditional money based on the degree to which they protect the privacy and preserve transactional ACID (atomicity, consistency, isolation, durability) properties. Finally we apply our evaluation criteria to two proposed electronic commerce systems: Digicash, (Chaum, 1985; Chaum, 1992) a token-based system; and NetBill, (Sirbu, 1995) a notational system.

1 Token and Notational Money

As befits its central role in our market economy, money has been the subject of much consideration. As researchers build new electronic forms of money, it is important to keep fundamental monetary properties in mind. This paper examines these properties for both traditional and electronic commerce.

This material is based on work supported by the National Science Foundation under cooperative agreement IRI-9411299. Additional support for Camp and Tygar was received from ARPA contract F33615-93-1-1330, a contract from the US Postal Service, and an equipment grant from IBM. This work is the opinion of the authors and does not necessarily reflect the opinion of any funding sponsors or the US Government.

Money can be defined either in terms of the functions it performs, or in terms of its representative forms. Scholars generally agree that money serves three major functions. Perhaps most important is its role as a *medium of exchange*. By replacing barter with a two step process of selling one good for money, which is then used to purchase another good, exchange transactions are greatly simplified. As a *standard of value*, money units are used to measure the worth of different goods or services so that their value can be compared in terms of a single numeraire. Finally, money serves as a *store of value*. Wealth may be stored more easily in the form of money than as cattle or loaves of bread.

Consistent with its function as a store of value, many of the earliest forms of money consisted of valuable objects which people were willing to take in trade. Thus the gold in a gold coin may have a value as a commodity comparable to its monetary worth. Other valuable items which have been used as money include jewels, shells, arrowheads and other tools (Haddon, 1949).

Eventually, money in the form of an asset with intrinsic value yielded to the need for money convenient to transport, exchange, and store. Thus, the use of tokens having no intrinsic value became common. The first paper money in the United States was legally negotiable notes issued by banks which required payment on demand of an equivalent value of gold (Rubin, 1994). These notes had no intrinsic value, but were readily convertible to intrinsically valuable money.

Easily convertible notes imposed some of the same limits on economic growth as intrinsically valuable coins. Thus easily convertible notes yielded to the use of trusted paper. While today's dollar may still serve as a store of value, currency itself no longer consists of tokens with intrinsic value. Since 1971, United States

currency has not been convertible to gold (Rubin, 1994). The use of mere pieces of paper as a medium of exchange, rather than tokens with intrinsic value, rests on a foundation of trust among the public that these pieces of paper will continue to be accepted in exchange and will serve as a stable store of value. A loss of trust could lead to a rapid return to barter, as is seen in countries where hyperinflation has rendered paper currency worthless overnight.

While notes are considerably more convenient than gold coins, they are still difficult to exchange in large amounts, and transport or store securely. Currently, most money consists not of pieces of paper currency, but rather of notations in the ledgers of depository institutions such as banks. We refer to this as *notational* money to distinguish it from *token* money or currency. Exchanges based on notational money require the debiting of one party's account and the crediting of another party's account. Institutions accepting demand deposits are required by law to be prepared to convert these notational deposits into currency on demand, thus providing convertibility between demand deposits and currency. Today, the amount of money which exists solely in the records of depository institutions vastly exceeds the value of currency in circulation (Heggestad, 1993).

A variety of instruments are used to instruct a bank to transfer notational money between accounts; the most common is a check. A complex system involving the Federal Reserve as a clearinghouse supports check clearing when accounts are held at different banks. In the last several decades, instructions to transfer notational money between accounts are increasingly sent electronically: wire transfers, ATM transactions, or more generally, Electronic Funds Transfer.

Notational and token currency are boundary conditions. For example, cashiers checks are notational money, but they share some of the properties of token money. There are also interfaces between the two: ATM machines and bank tellers exchange notational currency for token currency.

2 Desirable Characteristics of Money

To realize its multiple functions, money must have several characteristics both in individual transactions and as a currency system enabling these transactions. In this section, we enumerate these key characteristics.[1] We use the computer science vocabulary to describe transactions.

Consider purchasing an item. This transaction should have four characteristics: atomicity, consistency, isolation and durability. These four properties are commonly used to describe computerized transaction systems and are referred to as the ACID properties (Gray, 1993).

- Atomicity: Either a transaction occurs completely or it does not occur at all. For example, consider what happens when I transfer funds from a savings account to a checking account. Either my checking account is credited and my savings account is debited or neither account balance changes. (Tygar, 1996)
- Consistency: All relevant parties must agree on critical facts of the exchange. For example, if I buy a good for three dollars, the merchant and I should both agree on the amount of the purchase. After the purchase is completed, we must agree on that fact as well.
- Isolation: Transactions should not interfere with each other, and the result of a set of overlapping transactions must be equivalent to some sequence of those transactions executed in non-concurrent serial order.
- Durability: Even if my computer or the merchant's computer crashes, we should be able to recover to the last consistent state. For example, money that was available to a computer before it crashed should not disappear when the machine reboots.

We can further subdivide atomicity into two cases: money-transfer atomicity and goods-transfer atomicity. In money-transfer atomicity, funds are transferred atomically. In goods-transfer atomicity, not only is money transferred atomically, but the goods are also linked atomically with the transfer of money. For example, if I pay a dollar for a text file but never receive that text file, then goods-transfer atomicity is violated.

Besides the ACID transaction properties, money has other desirable characteristics. The effort (and cost) associated with conducting a transaction with all the desirable properties should be low so as to make low value transactions economical. Also, as a medium of exchange, money must make possible both low value and very high value transactions. This is facilitated by *divisibility:* mechanisms which allow for the exchange of multiple low denomination instruments for a single high denomination instrument.

Monetary systems should also be *scaleable* in the number of users. Money systems must support many consumers simultaneously buying goods from many merchants.

[1] A recent working paper from the Cross-Industry Working Team for the National Information Infrastructure has produced a similar list of properties which they list under the following headings: stability of monetary value, exchangeability, retrievably, tamper-resistance. They also list as desirable properties of transactions: non-refutable, accessible, reliable, private, protected. See http://cnri.reston.va.us:3000/XIWT/public.html

The larger the community of users who trust and accept the particular form of money, the more that form of money's value as a medium of exchange is increased. Thus, a single national currency is preferable to a plethora of local currencies issued by regional banks. Similar considerations are driving the European Community to consider a single currency. Alternatively, there must be well accepted and relatively fixed mechanisms for converting among various forms of money. We refer to this property as *interoperability*.

Money should have temporal consistency as well as transactional consistency. For money to hold its store of value, it must not be possible for individuals to create or counterfeit money. An excess of money will lead to a loss of trust in money as a store of value and consequently as a medium of exchange. In this paper, we concern ourselves only with unauthorized creation of money (rather than restricting governments, banks, and other authorized institutions.) It should be easy to distinguish "authorized" money from illegitimate money (McClellan, 1995). Public trust that money is legitimate is an essential element in its successful use as a medium of exchange. (Heggestad, 1992; Rubin, 1994; Haddon, 1949)

Money must also have temporal durability as well as transactional durability. When we put money away for "a rainy day," we don't want it to fade away. Money should be easy to store and retrieve. Here we refer to temporal consistency and durability by saying that currency is conserved.

Having considered transactional properties of fund transfers, we now consider how information in different types of transactions is disclosed to various participants and observers. In any transaction, there are many pieces of information and at least two distinct parties. Among the pieces of information are the amount, date, time and location of the transaction; the identities of the parties; the nature of the goods being purchased; and even the demeanor of the parties. When any piece of information held by one party is hidden from some other party, that information is said to be *private*. When a party's identity is hidden, then the transaction is said to be *anonymous*. Transaction participants may want to control the level of privacy or anonymity. Different pieces of transaction information may be accorded different levels of privacy or anonymity: a piece of information may be easily observed, may have a limited range of values, may be observed only under certain conditions,[2] or be completely hidden.

When traditional (non-electronic) token currency is exchanged, there is typically limited anonymity. Merchants or observers can narrow the range of possible customers in a cash purchase: the customer's gender, race, age, and (potentially) social class; the time of the transaction; and the location of the transaction can be determined by simple observation. Disclosures to third parties are limited in practice by the relative difficulty of physical (as opposed to data) surveillance.

There are also a substantial set of legal reporting requirements. In the United States, for statutory reasons, any electronic commerce system should be able to:

- provide detailed transaction information under subpoena (12 USC §1829, Bank Secrecy Act);
- prevent disclosure to law enforcement except under subpoena (12 USC §3403, Financial Privacy Act);
- promptly report any transaction above $10,000 (12 USC §1829, Money Laundering Act);
- store a copy of any transaction above $100 (12 USC §1829, Money Laundering Act); and
- record any action on a joint account in the records of all parties (15 USC §1591, Equal Credit Opportunity Act).

A notational currency system that can create and disseminate credit information as a primary business purpose must also:

- provide copies of consumer records to that consumer at his or her request (15 USC §1681, Fair Credit Reporting Act);
- delete obsolete information (15 USC §1681, 42 USC §3608, 12 USC §1708);
- provide relief to consumers in case of an error (15 USC §1681, 42 USC §3608, 12 USC §1708);
- limit data dissemination based on the primary business purpose of the requester (15 USC §1681).

Any notational system which is used to transfer wages must also provide a record, without the possibility of deletion, of personal income, or assure that the employer is able to do so (United States Tax Code, Section 61). Any currency system which is accessed by a card or similar device must protect the consumer against fraud by assuming all losses over $50 resulting from the loss of this device (when informed of such a loss by the consumer) (15 USC 1693, 12 USC §3403).

These requirements are extended if an electronic notational currency system provides more than notational currency transactions. For example, a system which removes the need for consumer credit agencies needs to provide some equal or better form of fraud prevention than credit agencies' protective bulletins.

These reporting requirements are typical rather than exhaustive. For further discussion of the types of reporting requirements, please see the appendix.

[2]For example, law enforcement access to information is conditional upon the ability to obtain a warrant.

3 Case Studies: Traditional Commerce Systems

3.1 Cash: Physical Token Money

Cash is token currency. Cash offers both privacy and anonymity because a dollar does not contain information that can be used to determine its transaction history. Neither does the exchange of cash necessarily create a record including the identities of those involved. Cash transactions usually provide anonymity of the buyer but not the seller. The privacy of cash is limited by the potential for physical observation. The information available to different parties in a cash transaction is shown in Table 1.

Info. / Party	Seller	Buyer	Date	Amt	Item
Seller	Full	Partial	Full	Full	Full
Buyer	Full	Full	Full	Full	Full
Law Enf	None	None	None	None	None
Bank	None	None	None	None	None
Physical Observer	Full	Partial	Full	Full	Full

Table 1: Information Available to the Parties In a Cash Transaction

Here we consider an observer who is physically well-placed: behind or beside the purchaser with a clear view of both parties and the transaction. A bank employee or officer of the law can obtain all the information available to an observer. However, there are no bank or law enforcement records produced in a cash transaction. It is reasonable to assume that no bank employee or law enforcement officer observes most cash transactions. Therefore the information available to a bank or to law enforcement is limited by what they would obtain from written records. Reporting of some transactions is required by law, but these reports depend on the active cooperation of the parties involved.

Paper currency trivially fulfills most of the requirements for an ACID transaction, including money-transfer atomicity. However, partially because cash is physical, cash suffers from scale limitations in transaction size and distance. Large transfers, such as those between banks, are problematic. Also, goods-transfer atomicity fails, especially when the transaction is over a distance.

There are no limits to scale in the number of users of cash, except those imposed by limits on the number of bills. Not only are individual transactions isolated, but the system is also free from bottlenecks.

Clearly there are security failures in the form of counterfeit notes, but security is generally maintained by a time-tested work factor. The design of the bills is periodically updated to discourage counterfeiting. Systems-level failures in the paper currency system are prevented by risk-limiting regulation, federal depository insurance, and the sheer magnitude of the task of passing enough counterfeit currency to upset the entire system.

Cash requires some trust between users. If a bill is determined to be counterfeit, the holder of the bill is not compensated. The validity of a bill can be partially verified during the transaction by visual inspection. By accepting cash, merchants imply only that they trust their own ability to detect counterfeits.

Cash is divisible in that there are many denominations; a large amount can be broken into many smaller amounts, and many smaller amounts can be exchanged for a single large denomination.

Cash is interoperable because it is legal tender. In fact, the lack of interoperability of state currencies was a driving force behind the creation of a national currency

3.2 Checks: Physical Transfers of Notational Money

Checks simultaneously create a transaction and a record. This record includes customer identity, date, location, and amount of a purchase; and is available to all the parties in the transaction, including all banks and clearinghouses involved. In the United States, there is no legal constraint on the disclosure by banks of customer purchasing habits, except that information can not be improperly disclosed to the government[3]. Absent explicit customer notation, checks provide *content privacy*. Content privacy means only that the actual purchases made are never necessarily recorded by any third party; the existence, location and parties to a transaction might be known.

Banks' records of transactions must be available for law enforcement access for five years after the initial transaction. Banks, and therefore government, may obtain partial information on a purchase if it is so noted on the check. Banks offer credit information to direct marketers and potential employers. They also provide records to law enforcement under subpoena. However, the retrieval, aggregation, and distribution of data derived from checks is limited by the expense of image and handwriting recognition systems.

[3]This is a constraint on government receipt of information, not bank disclosure. Fourth Amendment protection applies to personal records held by banks.

A check is not a token that represents value; it is a contract that authorizes the exchange of money from one account to another. Checks are instructions to transfer notational currency between demand deposit accounts.

Table 2 shows the information available to all parties in a checking transaction. Again,, we consider a well-placed observer who can actually read the check. The assumptions about law enforcement from the cash discussion remain true. (We put table entries that differ from the cash, as shown in Table 1, in boldface.)

Info. / Party	Seller	Buyer	Date	Amt	Item
Seller	Full	**Full**	Full	Full	Full
Buyer	Full	Full	Full	Full	Full
Law Enf w/ warrant	**Full**	**Full**	**Full**	**Full**	None[4]
Bank	**Full**	**Full**	**Full**	**Full**	None
Physical Observer	Full	**Full**	Full	Full	Full

Table 2: Information Available to the Parties In a Check Transaction

Checks are money-transfer atomic; however, the status of a transaction is not clear for several days. Until a check has cleared or a stop payment order is in place, the check is in play and any transaction involving the check must be held open.

Checks are not goods-transfer atomic because of the length of time required to clear a check and for the same reason that cash is not goods-transfer atomic.

Checks are consistent in that both the merchant and customer agree on the amount of the check.

Checks are not isolated. The validity of a check, and the outcome of the checking transaction, may depend on another transaction. For example, my paycheck must be deposited for my rent check to be valid. Notice that if I have two cash notes, the fact that one is counterfeit does not affect the value of the other.

Checks are durable after final settlement. It is, in fact, straightforward to create notational currency that is more durable than paper currency. Duplicate notations can be stored at physically separate locations so that if a single physical location is destroyed, no money need be lost. Paper currency cannot be copied or reproduced for access at a separate location or a later time in the way that notational currency can. (However, paper currency also cannot be deleted by a program error!)

Checking transactions are not limited by size. The efficacy of a check in notational transfers, especially for large transactions, is limited by the need for trust. Checks depend on the credit of the issuer, and the validity of a check cannot be determined by examination. The holder of a worthless check, like the holder of worthless paper currency, loses.

Security in the checking system varies between merchants. Requiring proof of identity is standard practice. Some merchants require testament to credit worthiness such as credit cards. A fundamental security mechanism in the checking system is the existence of criminal penalties for check kiting.

The problem of interoperability (Kaufman, 1983) in checking accounts has not been solved; for example, a Pittsburgh family vacationing in Florida may be unable to cash a check even though demand deposits certainly are a form of money.

Checks are divisible in that they can be written for any amount. A check may be cashed or exchanged for multiple lower denomination checks.

In the checking system, scalability is provided by check clearing. Checks do require central processing; but we are far from the practical limit to scale in the number of users. The processing capability of banks is the limiting factor in the number of users in the checking system. However the distribution of liability has provided sufficient motivation for the banks to fulfill their processing obligations, and assures that when a bank acts slowly the checking system is not threatened.[5] In the United States, check clearing also uses the Federal Reserve, at regional check processing centers, which provide floats and processing. The price of these services explains part of the attraction of banks to electronic notational currency.

3.3 Credit & Debit Cards: Electronic Transfers of Notational Currency

With credit and debit cards, the instruction to debit or increment an account is made electronically. We include these cards in this section because, although they may use electronic communication, a physical device is required. The possession of this physical device is sufficient proof of the holder's identity and authority to authorize instructions on card's account. Point of sale (POS) systems accept credit or debit cards, and charge or debit the payer's account while crediting

[4]Law enforcement may have access to full information about a purchase if the merchant keeps records.

[5]If a payor's bank does not report to the payee's bank that the payor has insufficient funds in a timely manner the payor's bank becomes responsible for making good the check.

the payee. Other systems simply check to verify that a card is valid rather than immediately transferring funds.

Credit and debit cards create records similar to those created by checks, except the records are machine-readable. Some content information may also be recorded with credit and debit cards. Privacy and anonymity can be compromised by the ease with which this information is analyzed and distributed.

Conversely, credit cards can increase privacy by enabling remote transactions. Physical observers cannot monitor a secure remote transaction. Electronic communications can be monitored, however, by electronic observers. Remote transactions currently involve credit and debit cards, which create machine-readable records; but these transactions can theoretically be secure from casual observers. For our analysis, we consider an observer who is electronically well-placed. In this case, that means the observer can monitor transmissions between the customer and merchant. The observer cannot read encrypted information, but can read all other information.

POS systems provide detailed information about the transaction to merchants and associated financial institutions, as well as leaking information through physical observation to an observer. Because information is obtained by the bank, it can be available to law enforcement. Content information may be recorded by the card issuer, and content information in machine-readable form may be trivially obtained by the seller. Information distribution illustrated in Table 3 (changes from Table 2 are shown in boldface).

Info. / Party	Seller	Buyer	Date	Amt	Item
Seller	Full	Full	Full	Full	Full
Buyer	Full	Full	Full	Full	Full
Law Enf w/warrant	Full	Full	Full	Full	**Full**
Bank	Full	Full	Full	Full	**None**
Physical Observer	Full	**Partial**	Full	Full	Full
Electronic Observer	Partial	Partial	Full	None	None

Table 3: Information Available to the Parties In a POS Transaction

Here the "bank" can be replaced by any card issuer. Most POS transactions send information over phone lines; however, most POS transmissions (like ATM transmissions) are encrypted. Therefore, an electronic observer could only observe that there are communications between the store and the clearinghouse.

A credit card transaction is not money-transfer atomic, although it does appear atomic to the merchant. That is, the merchant is guaranteed payment by charging the merchant a percentage of every sale to cover the inevitable losses. From the customer's perspective, credit card purchases have a period in which payment can be canceled, either by the customer's request or by her not paying her credit card bills.

Credit card transactions are normally consistent in that the customer and merchant agree on the amount paid. (The case of Lyndon LaRouche is a counter-example to this.)

Credit card transactions are not isolated. There are cases in which a merchant obtains a block on user credit which is not always promptly erased. These can in some cases lead to failure of isolation.

Credit card transactions are durable. However it may take weeks for a credit transaction to clear.

A credit card transaction is limited only by the customer's credit limit. With ATM machines, there is a size limit for an individual transaction. This is to limit risk, just as the limit on currency denominations limits risk. However, this sometimes fails, as the recent theft of over $300,000 with a single card and access code illustrates (Wells, 1995). Like checks, credit cards can support a nearly limitless number of users.

There are limits to interoperability between credit and debit card systems. You cannot pay your American Express card bill with your VISA, except by first obtaining cash.

4 Case Studies: Electronic Commerce Systems

The discussion of credit cards illustrates how electronic systems can increase data availability and decrease anonymity. Electronic commerce may exacerbate this problem. We discuss two systems which are suitable for transactions over networks, where there is no physical card or token. Data surveillance is an issue in all electronic currency because electronic currency includes machine-readable records of transactions. Electronically-stored records can be retrieved and reviewed with greater ease than printed material. Ease of the transmission and distribution of records is increasing with the interconnection of electronic information systems. This tends to increase data aggregation, unauthorized access, intrusion, misuse, and disclosure (Compaine, 1988; Schoeman, 1992; Hochwald, 1993; Office of Technological Assessment, 1985; Davies, 1981; Fenner, 1993). The worst case in terms of data

disclosure is an insecure network with plaintext linked to a unique identifier and overheard by an electronic eavesdropper.

Conversely, electronic commerce offers a broader range of privacy options than traditional physical currency. Electronic commerce systems can protect anonymity while providing information necessary for law enforcement.

Electronic currency can offer perfect anonymity, or threaten data surveillance. Electronic currency offers a broader range of privacy options than physical systems. But the theoretically possible is not always practical. We will examine two proposed electronic commerce systems: Digicash (Chaum, 1985; Chaum, 1992), and NetBill (Sirbu, 1995). Both are designed to work in open network environments. As we shall see, the technical limitations of these systems, in some cases, conflict with the desired characteristics noted above.

We assume that all observers are electronic; that they are well-placed; and that can view all messages, but can not read encrypted messages without a decryption key.

4.1 Digicash: Electronic Token Money

David Chaum's Digicash system addresses the problem of data surveillance in electronic currency. In Digicash, parties receive electronic tokens generated by a bank. These tokens can be exchanged between two network users, much like paper currency can. In some ways, Digicash conforms exactly to the assumptions underlying cash, anonymous tokens issued by central authorities. A Digicash transaction is shown in Figure 1.

How does Digicash stack up against the legal requirements for reporting monetary transactions? It does no worse than cash. In particular, the enforcement of rules such as the requirement to report transactions greater than $10,000 continue to rely on the cooperation of merchants, as Digicash-using banks are unable to provide the necessary information. Digicash does pose a larger risk in one sense: with Digicash, it is very convenient to exchange large amounts of money. This could lead to more large value "cash" transactions subject to reporting. Conversely, Digicash also poses a smaller risk: like cash, tokens must eventually pass through the banking system. Since token validity needs to be confirmed by the bank, banks can flag any tokens valued over $10,000. Payers must voluntarily provide their identity to the merchant or the bank. Identification can be especially problematic in remote transactions.

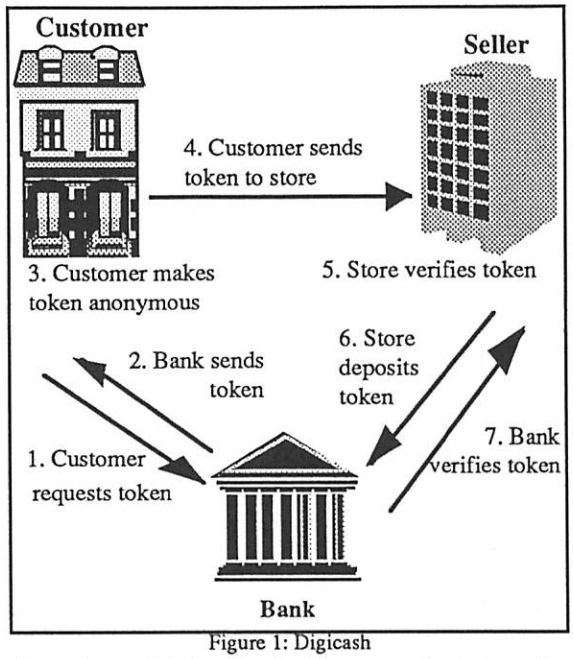


Figure 1: Digicash

How does Digicash stack up against the basic technical requirements for money expressed in Section 3? Digicash requires the use of a centralized server to act as an electronic bank. This creates a bottleneck, possibly limiting scale. On the other hand, there is no limit on the size of individual transactions, since it is easy to electronically transmit many tokens or to generate tokens that have high value.

Before presenting tokens to the bank for verification, the recipient modifies them to disguise the identities of the customer (previous holders of the token are already disguised.) In this way, Digicash provides anonymity: only the customer and the merchant know the identities of the parties engaging in a transaction.

There is a fundamental difficulty in the exchange of electronic tokens: electronic messages can always be trivially duplicated. Electronic currency cannot depend on physical means to make tokens difficult to counterfeit. To handle this problem, Digicash requires the merchant receiving an electronic token to quickly present that token to the electronic bank for verification, or to risk having a worthless token. This means that Digicash requires network reliability and availability to maintain integrity.

The information available to the parties in a Digicash transaction is shown in Table 4. (Items in boldface differ from Table 1, since cash is the corresponding physical currency.) Partial identity information trapped by the observer and the seller result from IP address transmission

Info. Party	Seller	Buyer	Date	Amount	Item
Seller	Full	Partial	Full	Full	Full
Buyer	Full	Full	Full	Full	Full
Law Enf w/warrant	**Full**	None	None	None	None
Bank	**Full**	None	**Full**	**Full**	None
Electronic Observer	**Partial**	Partial	Full	**None**	**None**

Table 4: Information Available to the Parties In a Digicash Transaction

Digicash's counterfeit-preventing techniques rely on several assumptions about the complexity of certain cryptographic. Under these widely accepted assumptions, Digicash is secure against counterfeiting.

Digicash also relies on assumptions about the privacy of cryptographic keys. These assumptions are problematic. In practice, it seems unreasonable to assume that cryptographic keys can always be kept private at electronic banks. Just as physical banks can not prevent occasional embezzlements, it seems reasonable to assume that Digicash banks can not always protect cryptographic keys. Thus, the question becomes: What damage results when the bank's private key is disclosed? An adversary who gains access to a cryptographic key can generate counterfeit tokens that are indistinguishable from valid tokens. These tokens can be generated in any amount desired, so compromise of the key compromises all tokens in circulation.

Digicash transfers are not money-atomic. (This was also noted by in (Yee, 94)) If a transfer of Digicash tokens (step 4 in Figure 1) is interrupted, then it is possible that both or neither party may believe it has legitimate access to the token. The customer may attempt to resolve this state by canceling the token (by cashing it in), but if the merchant also does this, the result is a race condition. (This also violates consistency and isolation). Since Digicash cannot reveal who cashed in a token when a merchant claims that he did not cash in the token, and the customer claims that the merchant did; dispute resolution can be a problem.

This could be resolved by having the bank log the identities of individuals and which tokens they have cashed in, but this would violate the Digicash anonymity model. The use of intermediaries could provide identity masking between buyer and seller, thus providing logging for the bank and maintaining anonymity. The result is shown in Table 5. Items in italics show the changes from Table 4 which result from using intermediaries; in boldface the information made available as a result of logging.

Info. Party	Seller	Buyer	Date	Amount	Item
Seller	Full	***None***	Full	Full	Full
Buyer	***None***	Full	Full	Full	Full
Law Enf w/warrant	**Full**	**Full**	**Full**	**Full**	None
Bank	**Full**	**Full**	**Full**	**Full**	None
Electronic Observer	***None***	***None***	Full	None	None

Table 5: Information Available to the Parties In a Digicash Transaction with Transaction Logging & Intermediaries

Digicash implements anonymous currency, but at the risk of duplication or arbitrary destruction of money. If Digicash were to implement durable transactions, then anonymity would be lost.

4.2 NetBill: Electronic Notational Money

NetBill is an electronic notational money system. In NetBill, parties write authorizations to exchange money analogous to checks. These authorizations must be presented to a bank; as with checks, they may need to be exchanged among several distributed banks. Each authorization is digitally signed and numbered, so counterfeiting requires key disclosure. See Figure 2 for an illustration of a transaction using NetBill. Steps 5-7 are digitally signed.

How does NetBill stack up against the basic technical requirements for money? NetBill uses distributed servers. It is an on-line system, and requires real-time response. Therefore, the integrity of the NetBill system depends on network reliability and availability and its scalability is limited.

On the other hand, because the system is distributed, it will tend to be robust and can be extended to handle arbitrary more transactions by using multiple NetBill servers.

NetBill transactions are fully ACID. In fact, NetBill satisfies goods-transfer atomicity for information goods because these goods are transferred encrypted. A cryptographic checksum of the encrypted goods is registered at the NetBill server after customer and merchant certification. The decryption key for the goods is registered with the NetBill server by the merchant. Transaction clearing is tied to the successful registration of the checksum and decryption key.

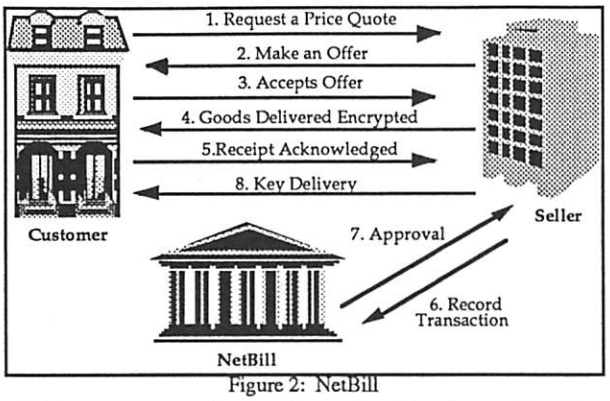


Figure 2: NetBill

If the customer does not receive the decryption key (step 8 in Figure 2) then the customer can request it from the NetBill server. If the goods have been misrepresented, a customer can dispute the transaction using the checksum of the goods and the decryption key registered at the NetBill server.

The integrity of NetBill rests on a number of complexity assumptions, as well as on assumptions about the security of NetBill servers. As with the case of Digicash, the former assumptions are generally accepted, but the latter assumptions should be carefully examined.

What damage will result if NetBill servers are violated? In the short term the server could move money between accounts. However, because NetBill and the parties involved log all transactions and have signed receipts, it would be possible to reconstruct bogus transactions.

Info Party	Seller	Buyer	Date	Amt	Item
Seller	Full	Full	Full	Full	Full
Buyer	Full	Full	Full	Full	Full
Law Enf w/warrant	Full	Full	Full	Full	**Full**
NetBill	Full	Full	Full	Full	**Partial**
Electronic Observer	**Partial**	**Partial**	Full	**None**	**None**

Table 6: Information Available to the Parties In a NetBill Transaction

How does NetBill stack up against the legal requirements for disclosing information about money? NetBill keeps a record of all transactions, so it is easy to comply with reporting regulations. NetBill normally provides no anonymity since users are identified to merchants and NetBill can trace transactions. The anonymity provided by NetBill is shown in Table 6. Since NetBill is an electronic notational system, the items in boldface are those that differ from a checking transaction. There are no legal constraints on customer and merchant data dissemination by NetBill ; or on the dissemination of data about customers by merchants. Boldface entries indicate a difference from Table 2 because NetBill transactions are based on the model of a check.

It is possible to boost the privacy of a NetBill transaction by using intermediary agents. Customer information can be transmitted through agents, which encrypt the origin of the message and then send it forward to the seller. The seller delivers the requested good through the agent. This is described in Figure 3 (Cox, 1994). It shows only one agent, but several can be used.

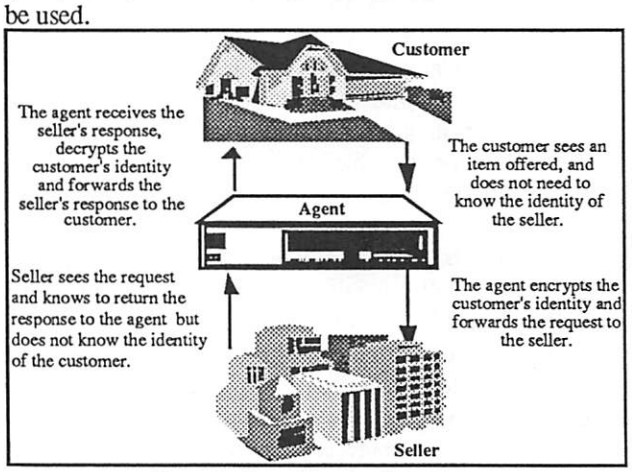


Figure 3. Use of Intermediate Agents

Mutual anonymity can be provided with the use of an additional agent for the seller. However, in all cases the NetBill server knows the identity of the parties and the amount of the transaction. Thus NetBill provides only limited privacy. (The NetBill server doesn't need to know the object purchased. This information can be

Info Party	Seller	Buyer	Date	Amt	Item
Seller	Full	**None**	Full	Full	Full
Buyer	**None**	Full	Full	Full	Full
Law Enf w/warrant	Full	Full	Full	Full	**None**
NetBill	Full	Full	Full	Full	**None**
Electronic Observer	**None**	**None**	Full	None	None

Table 7: Information Available to the Parties In a Privacy - Enhanced NetBill Transaction

simply hidden by any number of cryptographic techniques: public key encryption, private key encryption, hashing, etc.) The information known to parties in a privacy-enhanced version of NetBill is shown in Table 7, with the changes from Table 6 in boldface

5 Conclusions

All physical currency has innate anonymity losses due to the possibility of physical observation. Any merchant or observer can narrow the range of possible customer identities by simply looking at the customer! However, disclosure of information so obtained is limited by the relative difficulty of recording the data, and of physical, as opposed to data, surveillance. Thus in a cash economy, record creation can be costly and difficult. Many current government rules obligate parties in a cash transaction to create records so that the government can gather necessary information. Because data collection is inherently difficult, there was little need to put in place laws limiting data surveillance of transactions.

Modern electronic transactions transferring notational currency usually require the production of machine-readable records. Machine-readable records can easily be aggregated or distributed. These records are typically stored for extended periods, creating a hazard to privacy through inadvertent or deliberate disclosure.

As notational currency in the form of demand deposits, checks, and later, debit cards have become the predominant means of exchange, record creation has become more automatic and the resulting electronic records more prone to data surveillance. The government has taken advantage of the ease of record creation and processing with notational currency to impose additional record keeping and reporting requirements on banks and individuals. These records create a potential problem of data surveillance, and regulations to limit surveillance have not kept pace.

Participant anonymity in transactions involving notational currency is possible through the use of intermediaries, including anonymity of the user to the merchant, anonymity of the user to the financial services provider, anonymity of the merchant to the user, anonymity of the merchant to the financial services provider, and any combination of these (Low, 1993). Conversely, observation of transactions involving notational currency can be more difficult than for cash transactions, since face-to-face exchange is not necessary.

Electronic tokens may also provide seller anonymity, since these transactions no longer require a store front or postal address. Anonymity of the seller to the customer can provide perfect price discrimination. This is particularly useful for blind bidding, by simplifying the process and decreasing the opportunity for bribes and kickbacks. With this anonymity, companies could obtain discounts from reliable merchants (which could identify the customers) without the purchaser knowing the seller. However, without appropriate record-keeping this creates a fraud hazard.

Mutual anonymity between the service provider and the user is possible in both physical and electronic systems through the use of intermediaries. However, in physical transactions one must put full trust in the intermediaries not to alter the transaction or abscond with the cash, whereas in electronic systems it is possible through the use of cryptographic protocols to preserve anonymity and prevent transaction alteration by intermediaries.

New technologies, such as Digicash or NetBill, offer possibilities for increased privacy and anonymity in electronic monetary transactions. Digicash's promise of very high levels of anonymity are problematic, however, because such completely anonymous transactions are subject to statutory prohibition under various laws and regulations. We have also argued that meeting other technical requirements such as robustness would require record-keeping by participants that would also undermine the promised anonymity of Digicash.

6 Appendix: Reporting Requirements

Statutory reporting requirements have been developed to address problems of the intrinsic anonymity of token currency. These reporting requirements define the minimum auditing requirements for a currency system. In this appendix we consider both specific examples of reporting requirements and the general techniques used for reporting requirements.

Financial reporting requirements are necessary for a wide array of reasons, including tax collection, other law enforcement, social management, and tracing specific products (including stocks, weapons, and automobiles). Yet for all the differences in intent, these requirements use four basic data gathering techniques: immediate reporting, detailed record-keeping, periodic reporting, and periodic aggregate reporting. Following is an example of each.

An *immediate reporting* requirement was initiated by the 1988 Money Laundering Act which empowered the Treasury to require that all suspicious transaction be recorded. The Treasury interpreted this to require reporting of all cash transactions above $10,000 and all purchases of financial instruments (such as traveler's checks) over $3,000. All $10,000 transactions must be reported by all merchants, using the appropriate forms, to the Treasury. Whether this will apply to all businesses that accept electronic transfers of money has

yet to be determined; however, here we assume it will continue to apply to all consumer transactions. Currently a prompt report is legally required from all businesses, although the definition of "prompt" varies between types of businesses.

The Money Laundering Act extended the provision of the Bank Secrecy Act. The Bank Secrecy Act, despite its name, actually requires *detailed record-keeping*. It was passed to assure law enforcement access to detailed records of personal financial transactions under subpoena rather than privacy to patrons. The Bank Secrecy Act requires that financial institutions maintain records of all transactions over $100 for at least five years. Note the Bank Secrecy Act requires not only that records are in plaintext, but even requires that the record be an image of the bank's records of the transactions, such as a copy of the check (12 USC §1829d).

The most common *periodic data reporting* requirement is the annual individual tax filing on April 15. Wages, tips, and other forms of income must be reported to the federal, state and local governments as necessary for tax purposes. Expenditures can be reported according to taxpayer preference. That the increased record keeping possible in notational currency systems would be effective in preventing fraud is suggested by the thousands of dependents that disappeared from the tax roles as soon as their Social Security Numbers were required.

The Community Reinvestment Act requires *periodic aggregate data reporting*. The Act requires financial institutions make credit and depository services available to all the neighborhoods in their service area on an equitable basis. Regulatory bodies (the Comptroller of Currency, the Board of Governors of the Federal Reserve System, the Federal Depository Insurance Corporation, and the Federal Home Loan Bank Board The Comptroller of Currency, the Board of Governors of the Federal Reserve System, the Federal Depository Insurance Corporation, and the Federal Home Loan Bank Board) are responsible for specifying the necessary reporting requirements to verify compliance. Typically this means loan application aggregates sorted by ethnicity of the borrower, neighborhood, or loan amount. Specific data requirements vary over time, states, and even between institutions.

Statutes may prohibit as well as require reporting; for example, the Fair Credit Reporting Act (FCRA) placed statutory limits on disclosure. The FCRA prevents the disclosure of investigative consumer reports, and requires disclosure of credit reports only to those with a legitimate business need. The FCRA also requires disclosure to the consumer on demand. The FCRA requires accuracy in all records by empowering consumers to examine and correct their records. However, this offers limited protection because the courts have ruled that these limits apply only to credit reporting agencies, not to banks or insurance companies.

The Equal Credit Opportunity Act (ECOA) is another statutory limitation on data recording and dissemination. The ECOA also requires some disclosure and prohibits some data gathering. Specifically, the ECOA prohibits inquiries about martial status, thereby preventing the collection of this data. The ECOA also requires recording joint account activity in the records of all parties on the account. The ECOA applies to all companies that offer consumer credit and maintain credit records.

Public sector data gathering and disclosure is more tightly constrained than similar private sector activities. Disclosure of financial records of individuals obtained by the government is limited by the Financial Privacy Act and the Fourth Amendment. Data transfers between federal agencies are unregulated.

The previous examples are not an exhaustive set in terms of identification of reporting requirements; however, they illustrate the techniques used to obtain or prohibit data recording and disclosure.

Bibliography

12 USC §1829, Bank Secrecy Act

12 USC §3403, Fourth Amendment & Financial Privacy Act

12 USC §1829, Money Laundering Act

12 USC §1829, Money Laundering Act

15 USC §1591, Equal Credit Opportunity Act

42 USC §3608, 15 USC §1681, 12 USC §1708 Fair Credit Reporting Act

Chaum, D., 1985, "Security Without Identification: Transaction Systems to Make Big Brother Obsolete", *Communications of the ACM*, No 10, Vol. 28, pp. 1030-1044, October.

Chaum, D., 1992, "Achieving Electronic Privacy", *Scientific American*, Vol. 267, No. 2, pp. 76-81.

Compaine B. J.,1988, *Issues in New Information Technology*, Ablex Publishing; Norwood, NJ.

Cox, B., 1994, *Maintaining Privacy in Electronic Transactions*, Information Networking Institute, Carnegie Mellon University; Pittsburgh, PA.

Davies, 1981, *The Security of Data in Networks*, IEEE Computer Society Press; Los Angeles, CA.

Fenner, E, 1993, "How Mortgage Lenders Can Peek into Your Files", *Money*, pp. 44-48, April.

Gray, J. & Reuter, A., 1993, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers; San Francisco, CA.

Haddon, A.C., 1949, *A Survey of Primitive Money*, Methuen & Co. Ltd., London.

Heggestad, A., 1993, *Regulation of Consumer Financial Services*, Abt Books, Cambridge, MA.

Hochwald, L., 1993, "The Privacy Keepers", *Folio: The Magazine for Magazine Management*, Vol. 22, No. 12, pp. 62-63, 1 July.

H.M. Kaufman, 1983, *Financial Institutions, Financial Markets and Money*, Harcourt Brace Jonanovich, Inc.; NY, NY.

Low, S., Maxemchuk, N.F. & Paul, S., 1993, "Anonymous Credit Cards" *First ACM Conference on Computer and Communications Security*, November .

McClellan, D.,1995, "Desktop Counterfeiting", *Technology Review*, http://web.mit.edu/afs/athena/org/t/techreview/www/articles/feb95/mcclellan.html, February/March.

Office of Technology Assessment, 1985, *Electronic Surveillance and Civil Liberties OTA-CIT-293*, Office of Technology Assessment; Washington, D.C.

Rubin, L. & Cooter, R., 1994, *The Payment System: Cases Materials and Issues*, West Publishing Co.; St. Paul, MN.

Schoeman, F.D., 1992, *Privacy and Social Freedom: Cambridge Studies in Philosophy and Public Policy*, Cambridge University Press; New York, NY.

Sirbu, M., & Tygar, J.D., 1995, "NetBill: An Internet Commerce System Optimized for Network Delivered Services", *IEEE CompCom,*, San Francisco, CA; March 6, pp. 20 -25.

Tygar, J.D., 1996, "Atomicity in Electronic Commerce (invited paper)", to appear in *ACM/IEEE 21st Principles of Distributed Computing*

Wells, R, 1995, "Stolen ATM Card Nets $346,770", *RISKS* , Vol. 1, No. 83, pp. 21 February.

Yee, B., 1994 *Using Secure Coprocessors* . Ph.D. dissertation, Carngeie Mellon University. Available as CMU technical report CMU-CS-94-149

Economic Mechanism Design for Computerized Agents

Hal R. Varian
School of Information Management and Systems
University of California
Berkeley, CA 94720
hal@sims.berkeley.edu

Abstract

The field of *economic mechanism design* has been an active area of research in economics for at least 20 years. This field uses the tools of economics and game theory to design "rules of interaction" for economic transactions that will, in principle, yield some desired outcome. In this paper I provide an overview of this subject for an audience interested in applications to electronic commerce and discuss some special problems that arise in this context.

1 Mechanism design

As an example of mechanism design in action, let us consider the case of designing an auction to award an item to one of n individuals. Each individual i has a "maximum willingness to pay" or "value" for the item that we denote by v_i. We assume that this value is *private information* known only by person i. Our goal is to design an auction that will award the item to the person with the highest value.

This work was supported by the National Science Foundation Grant SES-93-20481. I wish to thank Jeffrey K. MacKie-Mason for his helpful comments on an earlier draft.

The most obvious way to do this is to use a standard *English auction*. In this game, the auctioneer continuously raises the price of the good. Bidders who are unwilling to pay the current price drop out until only one bidder is left. It is not hard to see that this remaining bidder must be the person with the highest value. However, it is important to observe that the price that he pays for the good will be the willingness to pay of the person with the *second* highest value (plus, perhaps, a tiny amount to break the tie).

This sort of auction is fine when communication costs are low and iteration is cheap. But what if communication costs are high? For example, suppose that one wants to conduct an auction that is distributed over space and/or time. Is there a way to achieve the result of the English auction without iteration? A standard form of one-shot auction is the *sealed bid auction*. In this game, each player submits a sealed bid. The bids are opened and the item is awarded to the person with the highest bid. That person in turn pays the price he bid for the good.

This auction avoids iteration but it will not in general achieve the desired ob-

jective of awarding the item to the bidder with the highest value. Suppose that bidder 1 has value of 1 and bidder 2 has value of 2. However, bidder 2 mistakenly *believes* that bidder 1's value is 1/2. Bidder 2 therefore bids $1/2+\epsilon$ and if bidder 1 bids any amount greater than this he will win the item.

Is there any kind of one-step procedure that will assign the good to the person with the highest value regardless of the accuracy of the beliefs of the participants? It turns out that the answer is "yes." The *Vickrey auction* works as follows. As before, each person submits a single sealed bid and the item is awarded to the person with the highest bid, but the winning bidder only has to pay the *second-highest* bid. (See [19])

It turns out that the optimal strategy in such an auction is for each person to bid his or her true value for the good. To see this, let b_i be the bid of person i and v_i the true value of person i. For simplicity suppose that there are only two bidders. Then the expected payoff to bidder 1 is

$$\text{Prob}(b_1 > b_2)[v_1 - b_2].$$

If the bracketed term is positive, then bidder 1 wants to make the probability term as large as possible. But if the bracketed term is positive, setting $b_1 = v_1$ makes the probability equal to 1, its maximal value. If the bracketed term is negative, bidder 1 wants to make the probability term as small as possible. But in this circumstance, setting $b_1 = v_1$ makes the probability 0, which is its smallest value. It follows that setting $b_1 = v_1$ is always an optimal strategy. Note that this is a *dominant strategy* in the sense that yields the highest expected payoff to each bidder *regardless* of the other bidder's strategy. Note further that the outcome is essentially the same as the outcome of the standard English auction: the highest bidder gets the item but he pays (essentially) the second highest price.

The Vickrey auction has been used in the computer science literature in [6], [15], [9], and no doubt in several other places. Since it is optimal for each person to reveal his or her true value, the Vickrey auction ensures that the item will be awarded to the person with the highest willingness to pay. However, it *does not* maximize seller revenue: that problem is considerably more complicated since the construction of the revenue-maximizing auction will typically depend on the beliefs of the seller about the buyers' values.

However, it is often the case that the auction that maximizes expected seller revenue has a form similar to that of a Vickrey auction. For example, if there are only two possible valuations for the good, then the seller should set a single take-it-or-leave it price if he believes that there is a high probability that the bidder has the high valuation and otherwise use a Vickrey auction. ([1], page 530.) [11] describes how the New Zealand government used a second-price auction for the spectrum with unfortunate results because they forgot to include this sort of "reserve price" requirement.

2 Computerized agents

The appropriate design of an economic mechanism depends critically on the model that one uses to describe the behavior of the participants. Economists have tended to use game theory to model participant interaction, although there has also been some work with evolution-

ary models.

Game theory has been justly criticized for its "hyper-rational" view of human behavior. However, such hyper-rationality may actually be an appropriate model for software agents: presumably software agents have much better computational powers than human beings. The whole framework of game theory and mechanism design may well find its most exciting and practical application with computerized agents rather than human agents, a point recognized by [15].

However, there are several additional considerations that come into play with artificial agents rather than human agents. First, to function effectively, a computerized agent has to know a lot about its owner's preferences: e.g., his maximum willingness-to-pay for a good. But if the seller of a good can learn the buyer's willingness-to-pay, he can make the buyer a take-it-or-leave it offer that will extract all of his surplus. Hence *privacy* appears to be a critical problem for "computerized purchasing agents." This consideration usually does not arise with purely human participants, since it is generally thought that they can keep their private values secret.[1]

Secondly, the artificial agent must guard against dynamic strategies that can extract private information. For example, suppose that an agent knows the lowest price at which its master will agree to selling the good (the "reservation price") and that it can safeguard this information from buyers. Suppose that this selling agent follows the simple strategy of accepting any offer that is higher than its reservation price. A buyer can then simply start at 0 and offer a sequence of incremental bids ensure that it ends up purchasing the good at a price slightly more than the seller's reservation price. This will typically not be a good deal for the seller!

This example is far from fanciful. In 1993 the Australian government auctioned off licenses for satellite-television services. The winning bid for one of the licenses, A$212 million, was made by a company called Ucom. Once the government announced Ucom had won, they proceeded to default on their bid, leaving the government to award the license to the second-highest bidder—which was also Ucom! They defaulted on this bid as well; four months later, after several more defaults, they paid A$117 million for the license, which was A$95 million less than their initial winning bid! The license ended up being awarded to the highest bidder at the second highest price—but the poorly designed auction introduced at least a year's delay into pay TV into Australia. See [11] for details of this story and how its lessons were incorporated into the design of the US spectrum auction.

In fact, the example shows why attention to mechanism design is important. If one can construct a mechanism for which truthfully revealing one's true willingness to pay is a dominant strategy, then there is no need to worry about keeping the willingess to pay private. The Vickrey auction is such a mechanism since the dominant strategy in this game is for each party to truthfully reveal the willingness to pay. A mechanism of this sort is called a *direct mechanism*. Somewhat

[1] Even if *current* information can be safeguarded, records of past behavior can be extremely valuable, since historical data can be used to estimate willingness to pay. What should be the appropriate technological and social safeguards to deal with this problem?

surprisingly it turns out that the class of direct mechanisms is much broader than it first appears. A fundamental result in the theory of mechanism design that we will outline below, *the revelation principle*, shows that anything that can be achieved by an arbitrary mechanism can be achieved by a direct mechanism. Hence the issue of keeping the willingness to pay private can be finessed if the mechanism is appropriately designed.

3 A Generalized Vickrey Auction

The Vickrey auction described above is a very powerful mechanism but appears to be of limited scope. However there is a generalization of the Vickrey auction that will handle much more complex problems—including many resource problems that appear to be quit different in nature. The Generalized Vickrey Auction (GVA) that I will describe below appears to be part of the mechanism design folklore, but it doesn't appear to be described in writing anywhere. Here I will provide a detailed argument, but I make no claims of originality except perhaps with respect to the exposition.

Suppose that there are $i = 1, \ldots, n$ consumers who each consume $j = 0, \ldots, k$ goods. Let x_i^j be the consumption of good j by consumer i. Good 0 will denote "money" and $x_i = (x_i^1, \ldots, x_i^k)$ will be the consumption bundle of goods by consumer i. Each consumer i holds some initial consumption bundle $\bar{x}_i$ and some initial amount of money $\bar{x}_i^0$.

An allocation $x = (x_1, \ldots, x_n)$ of goods is *feasible* if the total amount of each good held (including money) equals the total amount available:

$$\sum_{i=1}^{n} x_i^j = \sum_{i=1}^{n} \bar{x}_i^j,$$

for each $j = 0, \ldots, k$.

Each consumer i has a utility function $u_i(x) + x_i^0$; this is known as a *quasilinear utility function* and has certain properties that make it convenient for analysis. In particular, there are no "income effects" that influence the demand for the various goods. Note that this allows consumer i's utility to depend on the total allocation of the goods across all consumers not just on how much he gets of each good. In most of our examples, we specialize to the form where $u_i(x) = u_i(x_i)$, but in the last example we use the more general specification.

A reasonable objective in allocating the goods among the consumers is to allocate them in a way that maximizes the sum of utilities:

$$\max_{(x_i^j)} \quad \sum_{i=1}^{n} u_i(x) + x_i^0$$

$$\sum_{i=1}^{n} x_i^j = \sum_{i=1}^{n} \bar{x}_i^j \quad \text{for all } j = 0, \ldots n$$

In the simple case of the Vickrey auction described above, the utility functions were simply the difference between the value, v_i and the payment made by the consumer. Just as the consumer will not want to reveal his value to the seller, the participants in this resource allocation problem will not in general want to reveal their true utility functions. Our problem is to design a mechanism that will induce the participants to truthfully reveal their private information.

The Generalized Vickrey Auction

1. Each consumer i reports a utility function $r_i(\cdot)$ (which may or may not be the truth) to the center.

2. The center calculates the allocation (x_i^*) that maximizes the sum of the reported utilities subject to the resource constraint.

3. The center also calculates the allocation $(\hat{x}_{\sim i})$ that maximizes the sum of the utilities *other* than that of consumer i subject to the constraint that the allocation not use any of consumer i's resources.

4. Agent i receives the bundle x_i^* and receives a payment of $\sum_{j \neq i}[r_j(x^*) - r_j(\hat{x}_{\sim i})]$ from the center.

The final payoff to agent i in the GVA is given by

$$u_i(x^*) + \sum_{j \neq i} r_j(x^*) - \sum_{j \neq i} r_j(\hat{x}_{\sim i}).$$

I claim that if the GVA mechanism is used then it is in the interest of consumer i to report his true utility function $r_i(\cdot) = u_i(\cdot)$.

The first step in the argument is to note that the third term in the sum is irrelevant to consumer i's decision since it is totally outside of his control. To emphasize this we denote this term by K. It is useful in reducing the magnitude of the sidepayment to consumer i, but has no effect on the strategy of consumer i.

Next observe that the center will choose x_i^* to maximize

$$r_i(x) + \sum_{j \neq i} r_j(x)$$

subject to the resource constraint and consumer i wants them to maximize his payoff,

$$u_i(x) + \sum_{j \neq i} r_j(x) - K.$$

By inspection of these two equations it is optimal for the consumer to report $w_i(\cdot) = u_i(\cdot)$. This concludes the argument.

4 Examples of the GVA

Here we examine a few special cases of the GVA.

The Standard Vickrey Auction. In this case, the utility function of consumer i is $v_i - p$, where v_i is consumer i's value and p is the price he pays. Let $x_i = 1$ if consumer i gets the good and $x_i = 0$ if he does not. Then the sum of the utilities becomes

$$\sum_{i=1}^{n} v_i x_i$$

and the resource constraint is

$$\sum_{i=1}^{n} x_i = 1.$$

Of course x_i must be an integer, but this ends up being satisfied automatically so there is no need to impose that as an additional constraint.

Let m be index the consumer with the maximum value of v_i; then in order to maximize the sum of utilities the center will allocate $x_m^* = 1$ and $x_j = 0$ for all $j \neq m$. Let consumer s have the second-highest value; then if we eliminate consumer m the maximal sum of the remaining utilities will be v_s. The net payoff to consumer m in the GVA will be $v_m - v_s$ which is exactly the same as the Vickrey auction.

Multiple units of the good. Suppose that there is one good but there are $\bar{x}$ units of it to sell. Let (x_i^*) be the allocation that maximizes the sum of all consumers' utilities and let $\hat{x}_{j \sim i}$ be the amount allocated to consumer j if the sum of all consumers' utilities but consumer i

is maximized. Then consumer i's payoff is

$$u_i(x_i^*) + \sum_{j \neq i} u_j(x_j^*) - \sum_{j \neq i} u_j(x_{j \sim i}).$$

To see how this works, suppose that there are 2 consumers and 3 units of the good to allocate. Consumer 1 values the first unit of the good at 10, the second unit at 8 and the third unit at 5. Consumer 2 values the goods at (9, 7, 6), respectively. By inspection the optimal assignment is to give consumer 1 two units of the good and consumer 2 one unit of the good. Consumer 1 receives a total utility of 18 and consumer 2 receives a total utility of 9.

Here's how the GVA handles this problem. If consumer 1 isn't present, all the goods go to consumer 2 who receives a utility of $9 + 7 + 6 = 22$. In the GVA, Consumer 1's net payoff is

$$18 + [9 - 22] = 18 - 13 = 5.$$

So consumer 1 pays 13 for the 2 units of the good he receives.

Similarly, if consumer 2 isn't present, all the goods go to consumer 1 who receives a utility of $10 + 8 + 5 = 23$. Consumer 2's net payoff is then

$$9 + [18 - 23] = 9 - 5 = 4.$$

Hence consumer 2 pays 5 for the 1 unit of the good that he receives. The seller receives $13 + 5 = 18$ for the 3 units that he has sold.

Public goods. Suppose that each consumer i initially owns $\bar{x}_i$ units of the good. Consumer i can contribute x_i to a "collective good" (e.g., a pool for site licensed software) which will result in a total collection of $G = \sum_{i=1}^{n} x_i$. The sum of utilities is

$$\sum_{i=1}^{n} u_i(G) + \sum_{i=1}^{n} \bar{x}_i - G.$$

We assume that $u_i(\cdot)$ is a differentiable, increasing, concave function.

This is a classic public goods problem. The G^* that maximizes the sum of utilities satisfies the condition

$$\sum_{i=1}^{n} u_i'(G^*) = 1,$$

whereas the contribution that is optimal for each agent i acting on his own satisfies the condition

$$u_i'(G^\dagger) = 1.$$

Under the conditions we have assumed, the total voluntary contributions will be smaller than the socially optimal amount.

How does the GVA work to solve this problem? Let (x_i^*) be a pattern of contributions that maximizes the sum of utilities and let $(\hat{x}_{j \sim i})$ be the pattern of contributions that maximizes the sum of utilities omitting the utility and contribution of consumer i. The payoff to consumer i is then

$$u_i(x_i^*) + \sum_{j \neq i} [u_j(x_j^*) - u_j(\hat{x}_{j \sim i}].$$

To see how this works in practice suppose that there are three consumers each with an initial wealth of 10. If the total contributed to the collective good is $G = x_1 + x_2 + x_3$, consumer i will have a net value of $.4G - x_i$. The sum of the utilities over all 3 consumers is

$$1.2G + (30 - G) = 30 + .2G,$$

which is clearly maximized when $x_1 = x_2 = x_3 = 10$. The sum of utilities over any 2 consumers is

$$.8G + (20 - G) = 20 - .2G,$$

which is maximized when $x_1 = x_2 = x_3 = 0$. Hence the equilibrium payoff to consumer i is

$$[.4\times30-10]+[.8\times30-20]-[.8\times0-20] = 26.$$

Effectively each consumer must pay $.8 \times 30 = 24 - 20 = 4$ as a "tax" on top of the payment of 10 that he is already making. This tax represents the cost that the consumer is imposing on the other consumers through his presence. To see this note that if consumer 1 isn't present, the public good will not be provided. Therefore, it is the presence of consumer 1 that imposes a "cost" of $.2 \times 10$ on each of the other consumers.

This mechanism is essentially the celebrated Groves-Clarke mechanism ([5], [2]). In fact the proof of the GVA presented earlier is essentially the standard proof of Groves-Clarke result. (See, e.g., [18], p. 429.) The interesting fact is that this standard argument works for a much broader class of resource allocation problems than the classic public goods problem to which it is normally applied.

5 The Revelation Principle

The GVA is called a *direct revelation mechanism* since the "message" sent to the center is in fact the entire private information of the consumer: his utility function. One might imagine other "indirect" mechanism: the consumer announces a bid, or a reservation price. It is rather remarkable that anything that can be achieved by such an "indirect" mechanism can be achieved by a direct mechanism. This assertion is known as the *revelation principle.*

In this paper we have considered only mechanisms for which truth-telling is a *dominant strategy.* The revelation principle is valid under much more general circumstances but it is particularly easy to explain in this case.[2]

For notational simplicity let us index the different types of utility function by t so that $u_i(t, x)$ is consumer i's true utility if the outcome is x. Let r_i be agent i's reported utility type, let $r = (r_1, \ldots, r_n)$ be the set of all reports, and let $x(r)$ be the outcome if the reports are r. The function that assigns the outcome $x(r)$ is the mechanism. If truthtelling is a dominant strategy for each agent i then it must be the case that

$$u_i(t, x(r_1, \ldots, t_i, \ldots, r_n)) \geq u_i(t, x(r_1, \ldots, r_i, \ldots r_n)) \quad \text{for all reports } r_i$$

(This is called the *incentive compatibility constraint.*)

Let us now consider some other mechanism. Rather than just reporting the type t, this other mechanism allows consumer i to send some different message, m_i. If the consumers send messages $m = (m_1, \ldots, m_n)$ the resulting allocation is y. If m_i^* is a dominant strategy for consumer i

$$u_i(t, y(m_1, \ldots m_i^*, \ldots, m_n)) \geq u_i(t, y(m_1, \ldots m_i, \ldots, m_n) \quad \text{for all messages } m_i$$

What can consumer i's message depend on? It can't depend on the the other consumers' types since consumer i doesn't

[2] In fact, the revelation principle was first formulated for dominant strategy equilibria by [4].

know them. All that consumer i's message can depend on is his private information—i.e., his type. Accordingly, let us define a function $M_i(t) = m_i^*$ that gives the optimal message for consumer i if his type is t. By definition, $M_i(t)$ must satisfy

$$u_i(t, y(m_1, \ldots, M_i(t), \ldots, m_n)) \geq$$
$$u_i(t, y(m_1, \ldots m_i, \ldots, m_n)$$
$$\text{for all messages } m_i,$$

which is exactly the condition that characterizes a direct revelation mechanism. Since the optimal message only depends on the *true* type there is no loss in generality in designing the mechanism so that the message *is* the type.

In other words there is no loss of generality in restricting ourselves to direct revelation mechanism. This is very important for the design of computerized agents since it says in effect that there is nothing to be gained (or lost) by communicating anything other than the "essentials" of the problem.

Consider, for example, the auction problems examined earlier. Each consumer had an incentive to reveal his true value v_i; the auction design itself ensured that the consumer was not hurt by this full revelation. The fact that we can restrict ourselves to direct mechanisms makes the privacy issue alluded to before much less troublesome.

6 Brief introduction to the literature

The classic work that laid out the rationale and basic framework for the field of mechanism design is [7] . Useful surveys of mechanism design are available in [3]; [14] and [8] are particularly useful.

For interesting applications of mechanism design see [12], [13], [10], [11], and [20] for auction design. See [16] for matching models and [17] for price discrimination.

Several computer science applications of mechanism design that were influenced by the mechanism design literature are described in [15].

References

[1] Ken Binmore. *Fun and Games.* D. C. Heath, Lexington, MA, 1992.

[2] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–23, 1971.

[3] John Eatwell, Murray Milgate, and Peter Newman. *Allocation, Information and Markets.* W. W. Norton & Co., New York, 1989.

[4] Alan Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41:587–602, 1973.

[5] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[6] Bernardo Huberman and Tad Hogg. Distributed computation as an economic system. *Journal of Economic Perspectives*, 9(1):141–152, 1995.

[7] Leo Hurwicz. The design of mechanisms for resource allocation. *American Economic Review Papers and Proceedings*, 63:1–30, 1973.

[8] John O. Ledyard. Incentive compatibility. In John Eatwell, Murray Milgate, and Peter Newman, editors, *Allocation, Information and Markets.*

W. W. Norton & Co., New York, 1989.

[9] Jeffrey K. MacKie-Mason and Hal R. Varian. Pricing the Internet. In Brian Kahin and James Keller, editors, *Public Access to the Internet.* MIT Press, Cambridge, MA, 1995.

[10] Preston R. McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[11] John McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8(3):145–162, 1994.

[12] Paul Milgrom. Auction theory. In Truman Bewley, editor, *Advances in Economic Theory, 1985: Fifth World Congress*, pages 1–32. Cambridge University Press, 1985.

[13] Paul Milgrom. Auctions and bidding: a primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.

[14] Roger B. Myerson. Mechanism design. In John Eatwell, Murray Milgate, and Peter Newman, editors, *Allocation, Information and Markets.* W. W. Norton & Co., New York, 1989.

[15] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter.* MIT Press, Cambridge, MA, 1994.

[16] Alvin E. Roth and Marilda A. Oliviera Sotomayor. *Two-Sided Matching.* Cambridge University Press, Cambridge, England, 1990.

[17] Hal R. Varian. Price discrimination. In Richard Schmalensee and Robert Willig, editors, *Handbook of Industrial Organization.* North-Holland Press, Amsterdam, 1989.

[18] Hal R. Varian. *Microeconomic Analysis.* W. W. Norton & Co., New York, 1992.

[19] William Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[20] Robert Wilson. Auction theory. In J. Eatwell, M. Milgate, and P. Newman, editors, *The New Palgrave.* MacMillan, London, 1987.

Can the Conventional Models Apply? The Microeconomics of The Information Revolution

Bruce Don, *RAND*,
<Bruce_Don@rand.org>,
and
Dave Frelinger, *RAND*
<Dave_Frelinger@rand.org>

Operating with incorrect assumptions concerning information firms and how they conduct commerce has significant public policy implications. One possible consequence is inappropriate anti-trust action (or inaction) by government regulators. Because the decision to enforce is essentially Boolean in nature and can have long-term impacts on the industry, it is important to base regulatory and other public policy decisions on appropriate models.

This paper argues that there are importantly different microeconomic paradigms applicable to information-based commerce. These differences should be investigated in some depth to inform future policy decisions affecting information-based enterprises and economies based on their commerce. Such research may lead to significant improvements in our ability to make good public policy decisions on issues that will challenge us as our society adapts to the changes in commerce brought about by information technology.

Almost daily our society must confront issues that require us to apply analytic tools to information-based commerce in order to reach policy decisions that can have long-term effects on the industry and our economy—for example the question of information firms and the exercise of monopolistic power. It appears that there may be importantly different microeconomic paradigms applicable to information-based commerce. Since there are significant policy implications if we make decisions based on incorrect models or assumptions about information firms and their commerce, we should be uncomfortable in deciding whether we can rely on a free market approach, or if market failures call for a government role until we have done some research into how microeconomic theory applies to the information age. We currently understand the changing role of information in economics to such a shallow extent that we often cannot even agree on apt names for some of the mechanisms that we puzzle over. This level of understanding suggests that taking a careful look at the "microeconomics of the information revolution" at an early stage in its unfolding history may lead to significant improvements in how well informed our public policy decisions are on key issues.

To understand the need for this research in microeconomic theory we first need to understand how information technology is changing the role that information plays in our commerce, how important this may be in our economy in the future, and how this shift to information-based commerce poses a challenge to regulatory and other policy decisions because our analysis tools are limited. We start with how information technology is changing the ground rules.

Change in Information Technology and Its Impact on Regulatory Policy

From the standpoint of a policy-maker or policy analyst, the development and adoption of the underlying tools and processes that facilitate an information-based economy is occurring much faster then the regulatory process is equipped to handle. Multiple protocols are being deployed, tested, adopted, and in turn discarded at a pace that is staggering. Indeed when one considers just the recent exponential growth of the use of the World Wide Web and the Internet, it is almost impossible for governments to deal effectively with the issues surrounding such a rapid adoption of a set of tools, or of the variety of uses of those tools. If this were not challenging enough, the problem is further amplified by the fact that the micro-economic and policy-analytic tools, models, and theory we rely on in determining when market failures call for government intervention may be ill suited to analyzing commerce in the information age. The pace of change is not the only problem—the changing role of information in commerce is more fundamental.

The Role of Information in Post-Revolution Commerce

Information plays a number of roles in economic endeavors. These have long evoked a broad literature in microeconomics dealing with such aspects as price (as a form of information summarizing demand and supply) and investment in research (buying new information).[1] However, the thrust of this past thinking has shed only scattered light on what may be the more important aspects of the future role of information in economics, because it has yet to agree on a paradigm that regards information as product, or one that explains how information may be used as a

[1] For an excellent overview and synthesis of this literature see Priest, 1985 and 1994.

production input—roles which information increasingly plays, perhaps even dominantly plays, in our commerce.

The information revolution has seen an explosion in the number and types of enterprises engaging in such information-based commerce. Commerce based on informational goods such as data, text, software or interactive games, digitally recorded music, and video productions is notable because for firms producing these goods, the marginal cost of producing a copy for the n^{th} customer is essentially zero.

Not only does this imply that entertainment and software companies might find business profitable (if they can protect their property rights), but it may also imply that the traditional microeconomic tools we rely on for most analysis must be applied with great care when assessing information-based commerce. Additionally, if a substantial proportion of the firms in our economy deal in such information-based goods with near-zero marginal cost of production (they experience sizable increasing returns to scale over the relevant range), then general equilibrium becomes a less useful model of how such an economy would behave. Because these "increasing returns to scale" conditions more accurately describe complex adaptive systems than equilibrium systems, we might expect to see things like rapid, almost explosive growth of small companies, lock-in of the firms (and their technology) that first reached a critical size, and extinction (in contrast to adaptation) after they have outlived their era—particularly in the industries that deal with information-based goods.[2]

Additionally, firms that enjoy a near-zero marginal cost of production for their primary products (or near-zero cost of replication for their primary production processes) may have the potential for substantial monopolistic rents. Modern arguments that this is the engine that drives innovation and economic growth have put the role of information-based production and commerce in a new light as to their importance to the long-term growth of our economy.[3]

When considering the U.S. economy as a whole, the portion of the economy that is based to a large extent on informational goods is growing rapidly. Some sectors, such as the financial services industry, are already deeply involved in information-based commerce. In others, such as the entertainment industry, a transformation to information-based commerce is taking place. We are even witnessing the emergence of previously unknown forms of information-based commerce, as typified by the bio-infomatics industry. The aggregate contribution of these industries to the economy are increasing relative to man-power and capital intensive firms. Unfortunately, it is very difficult to reliably predict the ultimate growth during a time when entirely new industries are coming into being to satisfy demands for informational goods that have been heretofore non-existent.

Why Information-based Commerce Poses a Regulatory Challenge

A problem for governments, industry, and consumers is that many of the approaches to conducting and regulating commerce in physical goods developed over the centuries do not necessarily reflect the rules of this new and emerging form of commerce. Indeed, many of the implicit and explicit assumptions of legal and regulatory policies are based on what must now be considered suspect assumptions. The consequence of this is that there is an additional set of uncertainties for business concerning possible regulatory action, such as inappropriate anti-trust or other regulatory action (or inaction), and that this may jeopardize the international competitiveness of an important element of the U.S. economy.

A clearer picture of the problem is necessary to avoid some potentially serious problems that are likely to arise from piecemeal development of new "rules of the road." From the government's perspective doing nothing is not a viable option since the "rules of the road" for the existing economy are already codified and in place. The new information-based economy will have to be either consciously included or excluded from the already existing framework since it cannot be separated from the existing economy. If it is included, some adaptations will have to be made to avoid inappropriate applications of rules; if they are excluded, publicly acceptable rationales for exclusion will have to be made. In either case, understanding the nature of that new economy and the interaction of technology, social issues, and economics will be helpful to both the rule-makers and those affected by the rules.

Central to these quandaries is the concept of information-as-economic-good in contrast to the canonical view of information as a characteristic of

[2] Arthur (1990) has provided what are perhaps the deepest insights about the characteristics of such an economy using computational experiments.

[3] This perspective has a long history in economic thought dating back to Schumpeter (1947). Recent insights as to the role that the near-zero replication costs of knowledge- and information-based products and processes may play in our economy have perhaps been most fully developed by Romer (1994).

the market environment that influences decisions. For lack of agreed-upon terminology it is convenient to call these goods "informational goods" (in distinction to "physical goods").

An Illustrative Examination of Informational Goods

Since we do not have agreed-upon definitions of these "new" goods, an illustrative model would be helpful to better define the importantly distinguishing characteristics of commerce in informational goods.[4] This would give us a clearer picture of how pervasive such commerce is in our economy, and which firms might be subjected to policy decisions based on inappropriate analytic tools. Developing such a model also serves to illustrate what it might mean to investigate the "microeconomics of the information revolution."

A Model to Accommodate Informational Goods

Two importantly defining characteristics of economic goods are the property rights enjoyed by their producers and their consumption character. Because so much of our experience deals with physical economic goods, these characteristics are often thought of as being irrevocably bound together forming two categories: public goods and private goods. But once we contemplate trafficking in informational goods such a model appears too limited.

To describe some of the fundamental differences between a model that can accommodate both physical goods and informational goods and more limited models, it is instructive to rely on an illustration similar to that developed by Ruggie [1972, p. 888] which is shown in Figure 1. In the figure, economic goods are classified according to the degree to which they possess two characteristics: rivalry and exclusivity. The first, rivalry, describes the process by which the good is consumed. This in turn says something about the process by which the good must be produced or supplied to the consumer. At one extreme of the spectrum which describes this characteristic, we say a good is rival in consumption. This means that one's consumption of a measure of that good precludes the consumption of that same measure by another. Probably the simplest example of such a good is food; once you have eaten it others cannot. Other examples include television receivers, computer hardware and pre-recorded video tapes. To supply this good to consumers, a producer must create one measure (unit) for each customer.

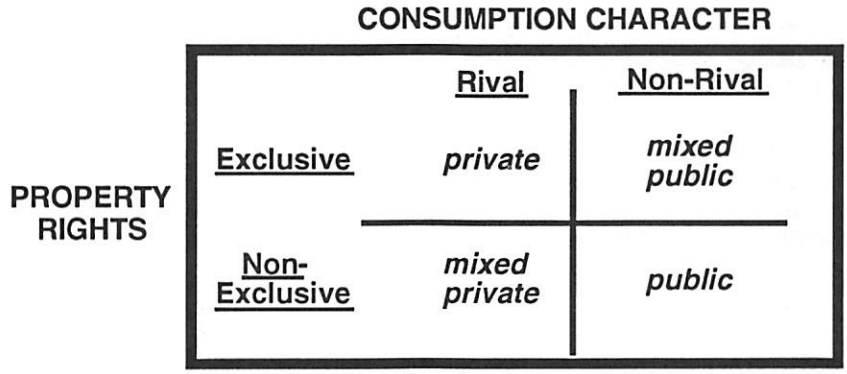


Figure 1 – A Model to Accommodate Informational Goods

At the other extreme of this spectrum are goods which are non-rival in consumption. These goods have the opposite characteristic—an individual's "consumption" of the good does not preclude the "consumption" of the same good by another individual.[5] A broadcast television program might be a simple example of a good with this characteristic. To supply this good to

[4]Of course, informational goods are not really new at all, but their prevalence and importance in our commerce, particularly in digital or electronic form, is both new and growing.

[5]History has shaped the economist's term "consumption" in the context of physical goods, and it lacks specificity when we attempt to distinguish between that consumption attributed to a rival good, in which consumption by one precludes consumption by another, and the consumption of a non-rival good, in which consumption by one does not affect the amount of the good available to others. We have used the term "enjoy" when referring to consumers extracting the utility from the latter.

consumers, a producer can create one measure of it, and service all customers who elect to enjoy it.[6]

The second characteristic, exclusivity, describes the property rights which characterize a good. This in turn provides insights as to how both judicial and physical laws have acted to determine what costs are necessary to protect these property rights by excluding other consumers. At one extreme are goods which have property rights which are said to be exclusive. By this we mean that it is possible (it costs little) to preclude consumption or enjoyment of the good by others. Music played on a cassette tape player provides a suitable example of this type of good. The person who owns the tape and player can exclude others' enjoyment if he or she so chooses—by using earphones. It should be noted that the owner's enjoyment need not prevent others from enjoying (or dis-enjoying) the music; it is merely a matter of the owner exercising his or her property rights. Goods with such clearly defined property rights can be sold and may form the basis for a market.

At the other extreme lie goods which have non-exclusive property rights. It is not possible (or very costly) to preclude others from consuming or enjoying such goods. A public beach or park are the classic examples. Because of the way we have defined property rights in such a situation, law-abiding citizens cannot be excluded.[7] But "free ware" also falls in this category—as does other freely copied software. Because consumers are not likely to buy goods which are available whether or not they pay for them, it may not be possible to sell these goods at a price which reflects their true value to the consumer or cost to the producer, so typically these goods must be supplied by benefactors or public agencies even in a free market economy.[8] We often think of information as falling in this category (which it can if its property rights can not be protected), but this is largely the result of using a model in which property rights and consumption characteristics are bound together as they are in physical goods.

Although we have examined the extremes of the characteristics illustrated in the figure, we have really described only two—*purely public* and *purely private*—of the four types of goods this model allows. This is because our discussions have focused on the joint extremes, "exclusive and rival" and "non-exclusive and non-rival."

Although these joint extremes often occur together, there is no universal requirement that they do so. For example, goods with non-exclusive property rights which are rival in consumption are also possible. They are termed mixed private goods in the figure. These goods are somewhat more difficult to typify. However, laws define theft and ultimately property rights, so they can also mandate that consumers must share with others the goods they have purchased.[9] Because these goods are rival in consumption a producer must create one measure for each customer.

Such a Model Does a Better Job of Accommodating Much of Today's Commerce

When considering how information fits in post-revolution economics, the most germane category is the last—goods with exclusive property rights which are also non-rival in consumption (termed mixed public goods in the figure). A rock concert typifies such goods, or better yet, software packages or a program on pay television. Because these goods are non-rival in consumption a producer can produce (author or create) one measure of such a good and service with copies for

[6]Over the relevant range of customers, for in real life crowding and transportation costs often limit "all" to a finite number.

[7]While our discussion here focuses on property rights, we should note that, in the real world, other aspects intervene to complicate the matter. For example, while there may be no legal barrier to one's access to a beach, the distance one must travel to enjoy it can significantly limit access. Our beach or park are only good examples provided the cost and effort of getting there is roughly the same for all the users we are considering.

[8]The case of copied software illustrates the need for more fully thought-out models. If the software is copied by someone who would not have purchased it, the producer has not actually suffered a loss (as the act of copying did not diminish his stock of information or preclude a sale). In fact, the situation may prove beneficial to the producer if the copy serves as a convincing advertisement for the next version of the software. If the copied product is sold, the producer may still not suffer a loss if the pirated product is sold to a consumer who would not have paid the legal price.

[9]Usually the property rights issue does not come into play when dealing with goods which are rival in consumption—another's enjoyment of the good you paid for smacks of theft. But initial proposals for a deep sea bed mining policy such as that incorporated in the pre-1980 drafts of the Law of the Sea Treaty would have created such a good. The technology to commercially exploit the large quantities of potato-sized nodules composed of manganese, copper, cobalt and nickel had been developed by a few industrialized countries. The proposed policy would have required those countries exploiting the nodules to share their harvest with less developed nations incapable of exploiting those resources. The decision to pursue creation of such a good was an acknowledgment of the ambiguity of international law in regards to the sea bed, the difficulty and inconvenience in enforcing property rights in that environment, and of the relatively marginal value of those goods. Interestingly enough, a similar arrangement has also been proposed for lunar mineral rights, as well as by some for benefits arising from genetic sequences and products extracted from organisms indigenous to Third World nations.

all consumers who wish to enjoy it.[10] Because they have exclusive property rights, these goods can be sold at a price which reflects the consumer's valuation of the good, but not necessarily the producer's cost of production. The physical goods that most of our microeconomic theory was created to address are rival in consumption and because of this, our convention in law is that they bestow exclusive property rights on both the producers and consumers that traffic in them. Informational goods on the other hand are non-rival in consumption and have a mixed profile of property rights. Those that the law protects as having exclusive property rights, such as motion pictures protected by copyright, form the basis for some of our most lucrative markets (e.g., the entertainment and software industries).

Firms That Deal in Informational Goods

More specifically defining the characteristics of informational goods through the use of a model such as the one illustrated above leads to the conclusion that there are a number of types of firms that might be characterized as informational firms based on the nature of their products. Some of these (software firms) are what we might naturally think of as information firms, others (such as pharmaceutical manufacturers) do not so readily come to mind when contemplating information-based commerce, because they deal in more tangible products. Table 1 qualitatively illustrates the similarities and differences of these firms in terms of some of the characteristics discussed earlier in the paper.

Table 1
Firms Producing Informational Goods May Not Always Be Part of the "Information Industry"

Selected Characteristics	Software	Digital Movie / Entertain-ment	Tele-Communi-cations	Pharma-ceuticals	Bio-infomatics
Property Rights	Exclusive	Exclusive	Exclusive	Exclusive	Uncertain
Development Cost	Low / Moderate	Moderate / High	High	Moderate	Low
Physical Capital Investment	Low / Moderate	Low / Moderate	High	Moderate	Low
Consumption Characteristic	Non-rival	Non-rival	Non-rival when unsaturated Rival near saturation	Rival	Non-rival
Marginal Cost of Production (Nth Unit)	Almost nil	Low	Very low	Very low	Almost nil
Increasing Return to Scale	Yes	Yes	????	Yes	Yes

What has changed is not so much the type or characteristics of economic goods, but simply the prevalence and importance of informational goods in our economy and therefore the need for a model that is more explicit about the characteristics that distinguish informational goods from physical goods.

[10] But often only once to each consumer. In contrast to rock concerts or musical recordings, many information goods satiate the consumer with only one measure—once it is added to their stock of knowledge consumers have little desire to acquire it again, unless the information becomes old or is forgotten.

The **software industry** produces a wide variety of products for manipulating data and controlling devices. As already discussed, the actual physical products produced are very low in cost, while the value of the information is very high. Software firms are the archetypal information firm with normally low costs of entry for uncrowded sectors, but with increasing costs of entry as features are added to differentiate products, or as more functionality is desired.

The **movie and entertainment industry** is comprised of firms providing content (programming and movies) for broadcast or other

means of selling that content. These firms are able to exclude non-paying customers from viewing their products in non-broadcasting milieus, and to control to whom the initial release for broadcast is made. The marginal cost of delivering the product to the n^{th} customer is very low for the entertainment industry, but the production costs can be large enough to necessitate large numbers of consumers supporting the product. While not normally thought of as informational firms, the growth of alternative mechanisms for distribution of their product—digital data available from a server rather then physical items such as films and video-tapes—are beginning to force many of the same economic factors that dominate the software industry on to the movie and entertainment industry.

The **telecommunications industry** is in the business of transporting information from producers to consumers. Many consider the telecommunications firms to be information firms because of this association with information. However, unlike firms discussed earlier, this industry requires a relatively large capital investment to enter the game, and in general behaves like other common carriers when they are not overloaded.[11] The cost per additional individual user is nearly nil because of the large band-width available for communications relative to the small impact of an individual user. As band-width is required when the system is approaching saturation, it can be added in large increments, allowing for large numbers of additional users to be added.

The mainstream **pharmaceutical industry** is not typically thought of as being made up of information firms since it produces tangible goods (drugs and devices). However, the major characteristic of these firms is that the drugs that are developed have a very high development cost and a generally low cost of production for individual doses. For many of these firms, the development of information on the compact of different drugs is the firms' primary job.

As yet the **bio-infomatics** industry is in its infancy. The large-scale sequencing and manipulation of the data comprising the genetic structure of organisms is just now beginning to support bio-engineering tasks. This industry is distinct from that of the pharmaceutical industries in that the primary products are the information on the sequenced DNA itself, and the tools for manipulating that information, rather than the physical products they yield. One complicating factor for this industry is the uncertainty of rights to the data itself. The United States is currently allowing the patenting of applications using manipulated DNA. In this model, the expression of the sequence is protected, but the basic data is still viewed as a discovery that does not deserve patent protection. However, others are arguing that the sequence, whether or not it has a known function is patentable. Similarly, the copywriting of data does not appear feasible, but the indexing of that data and ordering of it in a database may be copyrighted. Some companies are positioning themselves to control a large fraction of the data as well as the tools for manipulating the data so as to make it unattractive for other firms to sequence DNA rather then simply purchase it.

Commerce Based on Information Goods

The firms that deal in informational goods are typically viewed as highly profitable. Does this re-thinking of what they are producing (e.g., the informational content of the CD-ROM, not the physical storage device) suggest any insights as to why this is so?

As already alluded to, digital and other technologies that allow information (music, video, programs, numerical data) to be easily reproduced move firms closer to working purely with information. This is in contrast to producing the physical manuscripts that were used to record and store information on in the past (e.g., books, video tapes, etc.) Making a copy of information held in digitized form neither detracts from the creator's original stock because information is non-rival in consumption and takes little time or effort to produce the serving for the n^{th} customer. The result is that the marginal cost of production is essentially zero.

If the property rights to what is being produced are exclusive (to the producer), the firm has a market. If the marginal cost of production is very low, the firm has the classic characteristics of a natural monopoly because it exhibits increasing returns over a very large range (essentially all potential consumers in the market). This has several significant implications.

First, there is substantial room for monopolistic rents—in fact one could argue that

[11] There is a difference between wireless communications modes and terrestrial fiber-based modes of transmissions. For wireless systems, for example cellular phones, the issue is the availability of band-width within a limited portion of the frequency. Physics limits the amount of communications that can be forced into a given portion of the radio-frequency spectrum, and aside from utilization improvements operators have to live with the limitations of nature. For terrestrial fiber based systems, additional fiber bundles can be easily added along existing rights-of-way allowing for increases in capacity.

this is the incentive that drives the innovation and economic growth that seems so prevalent in information-based commerce today.

Next, such advantages may accrue to broader segments of the economy than we commonly recognize, as the informational content of goods outweighs their physical content. In the micro-manufacturing and magnetic disk recording-head industries for example, the marginal cost of producing the physical goods is quite low—most of the value of the product comes from the information embodied in the design of the product. Firms operating under these conditions may enjoy advantages similar to firms producing informational goods.

Finally, we may be poorly characterizing an important factor of production—information. A long-term, historic view could argue that at first the production function included only labor; then added land; then capital equipment; and finally—and only recently—information. Indicators of the importance of information in the production function are available. Private equipment investment in computers and other hardware-based information technology appears to have surpassed industrial machinery investment in the early 1980s. However, a direct measure of capital investment in information itself is much more difficult to make because of data and accounting limitations.[12] Information-based capital investments such as desk-top publishing and film-editing programs have unique advantages over more traditional production equipment given their ability to be used over and over again in the production process without wearing out. If we do not include information in the production function, and it is actually an important input, we will attribute any productivity associated with information to the other factors of production.

Our Models and Information-based Commerce

With a potentially large number of the firms in our economy engaging in wide-spread information-based commerce, and a concern that this has important differences from the physical-goods commerce that current models have been tailored for, we should seek a clear understanding of just how well our current models work in this new venue. To this end we need a careful inventory of our microeconomic analysis tools to understand where they apply "as is," where they need to be modified, and where they are inappropriate or limited. A cursory look highlights some limitations that may result in important shortcomings:

- Despite the fact that we are willing to pay quite different prices for yesterday's and today's commodity trading data, or that seven megabytes of office software can command a price five times as much as a similar measure in game software, the current application of microeconomic theory often treats information as an undifferentiated good. (Vintage and quality of information matter to both consumers and producers.)

- We have problems with definitions—we use some information as factors of production (e.g., input data-sets); other information as part of the production process itself (e.g., word processors); and still other information as final products (e.g.. motion pictures). However, these "flavors" of information are not commonly defined, or distinguished, so it is difficult to say that we fully understand what we can not as yet even name. (A sense of verb and noun is involved in the role information plays in economics as well as familiar economic characterizations, such as intermediate goods and capital goods, which are normally only attributed to physical goods.)

- Counting poses a problem. Books, software packages, and megabytes are all seemingly ready measures. When we packaged our information in familiar, often physical containers (video cassettes, books, broadcast TV packaged in time slots), we had a clear understanding of what we were measuring. Nor did our economy cause us to worry much about comparing the quantity of information sold in book form to that, for example, sold in software package form. But, as we move toward trafficking in information in purely digital form, a fundamental characteristic about information becomes more readily appreciated—sharing information is not like sharing physical goods. A second user's use and enjoyment of information ("consumption" in canonical terms) does not necessarily diminish the first user's ability to enjoy it. (Rivalry in consumption and property rights combine differently than in physical goods.)

These are only a few examples, but they serve to warn us that our analysis tools can break down—or may be extremely difficult to use properly. Since we use these tools as a way to structure our thinking about such problems as whether or not we are faced with a market failure

[12] For example, in some firms software expenses under some threshold (e.g. $1000) are not considered capital expenditures even though the aggregate expenditures on these software packages may be significant and the re-capitalized (upgraded) software has a useful life longer than the hardware it runs on.

and need government intervention or regulation, this has substantial implications for the policy choices we are increasingly challenged with as the "Information Revolution" unfolds.

Implications for Policy Choices Based on Microeconomics

What are the implications if there are many firms in an economy that produce informational goods and our current models of how information-based commerce works have problems? First, the classical theory of the firm may be largely inappropriate even for small firms which, rather than being "price-takers" in the market, have the potential to grow quickly from garage-based operations to lodge themselves in large-market-share positions. Because of this, other microeconomic models of the firm may provide better descriptions of what goes on than price-taking behavior does, and our antitrust approach to firms that exhibit "monopolistic behavior" may need to be re-thought, especially in economies which seem to be driven by technological growth.

Next, since we currently do not specifically include information in the production function, we may not adequately understand technology-driven economic growth, particularly if many of the firms in our economy use information as a factor of production in either a consumable (time perishable input data set) or a capital investment (a long-lasting software package for product design) sense. As a result, tax incentives may emphasize the wrong kind of investments—physical capital in contrast to informational capital.

Finally, if both factor markets and product markets are characterized by large segments which enjoy increasing returns to scale (because they deal in informational goods) then conventional general equilibrium becomes harder to apply, and the economy that results is best described as a complex adaptive system. If the economy displays substantial increasing returns to scale, we should expect to see things like rapid, almost explosive growth, lock-in of firms and their technology, and extinction (in contrast to adaptation) as more commonplace in our economy than heretofore. If our model of a firm's life cycle bespeaks many firms experiencing rapid growth to substantial market share and rapid extinction because of the market forces they face, our approach to government rescue efforts, like that for Chrysler, and the timing of antitrust actions, like the 1980s investigations of IBM, will need to be viewed in a different light.

A Need For Investigation

It appears that there may be importantly different microeconomic paradigms applicable to information-based commerce. While many of these have been reflected in case study analysis of industries not generally regarded as engaging in information-based commerce (for example the pharmaceuticals or video rental industries), they have not been applied as broadly as needed. Since there are significant policy implications of making decisions based on incorrect models or assumptions concerning information firms, forging an understanding of the "microeconomics of the information revolution" at an early stage in this unfolding history may lead to significant improvements in how well informed our public policy decisions are on key issues.

We are already confronted by such issues—for example the question of information firms and the exercise of monopolistic power—on almost a daily basis. Further, a determination whether or not such a "market failure" is, in fact, a problem that requires government intervention or regulation is indelibly colored by how we (as a society) think about the issue—the problem structure, theory, and models we apply. Because poor policy choices can have long-term effects on the industry and our economy, it is important to develop some clearly thought-out foundation upon which to base regulatory and other public policy decisions. Recognizing that we currently understand the changing role of information in economics to such a shallow extent that we do not even have apt names for some of the mechanisms we puzzle over, it would be prudent to be skeptical that any current assessments are based on a sufficient understanding of information-based economic interaction.

Balancing the implications of such a simple view are a host of factors that still need to be accommodated. It may be that the operation of the "unseen hand" is not that different after all—or that market forces, imperfect as they are in this new venue, may still do better than regulatory policy in keeping up with change in information-based commerce. For example, it may be that since consumers have little use for information that they already know, informational goods are so differentiated (each package of information being regarded differently by each consumer) that producers do not enjoy increasing returns to scale to any great extent. In yet a different vein, evolving software technology, such as search engines or abstracting software may largely displace whole segments of high value-added commerce (such as indexing and abstracting services) with appropriate gains in productivity for

those that use them (by lowering the transaction costs for acquiring the right intermediate informational goods). It may even be that such technologies allow market interactions that are not practical today, for example, we are already experimenting with pay-per-use paradigms in contrast to the pay-per-copy framework we usually use for informational goods today.[13]

In short, we should be uncomfortable in deciding whether or not we can rely on a free market approach, or if market failures call for a government role, until we have done some research into how microeconomic theory applies to this new age.

[13]This has the potential to overcome a basic paradox that causes difficulties in markets for informational goods. Purchasers do not know if the information that they wish to buy is useful to them until they buy it. With pay-per-use they may pay a small premium to see if it is useful, but what they pay would be largely determined by how useful they find the information.

References

Arthur, W. Brian, "Positive Feedbacks in the Economy," *Scientific American*, pp. 92–99, February 1990.

Priest, W. Curtiss, *An Information Framework for the Planning and Design of Information Highways,* Center for Information, Technology and Society, February 1985, with revisions September 1985 and October 1994.

Romer, Paul, "The Origins of Endogenous Growth", *Journal of Economic Perspectives*, 8(1), pp. 3-22, Winter 1994.

Ruggie, J.G., "Collective Goods and Future International Collaboration," *American Political Science Review*, 66(3), pp. 874–893, September 1972.

Schumpeter, Joseph, *Capitalism, Socialism, and Democracy*, New York, Harper & Row, 1947.

Internet Information Commerce: The First Virtual (TM) Approach

Darren New
<dnew@fv.com>
Senior Design Engineer
First Virtual Holdings Incorporated

More info at http://www.fv.com or info@fv.com

Talk Outline

I. The Problem: Doing Business on the Internet
II. Is Encryption the Solution?
III. Why Information Commerce is Special
IV. The First Virtual Information Commerce Model
V. How It Works
VI. What It Costs
VII. The Protocols
VIII. The Corporation
IX. The Future
X. What We Have Learned

I. The Problem: Doing Business on the Internet

The Internet is BIG. Really, really BIG.

Growing too fast to put numbers on my slides.

Unrestricted commercial use, enormous visibility.

We wanted to sell jokes on the net. (Really!)

Until October 15, 1994, no workable payment infrastructure

Traditional Approaches to Internet Commerce

-- Focus on goods and services
-- Some require encryption
-- Some require a closed system

First Virtual focuses on information commerce

-- "Information" means "delivered via the net"
-- No encryption required
-- Permits rapid introduction of commerce
-- Consistent with today's Internet

First Virtual's Goals

Focus on Information Commerce

Allow anyone to buy or sell

Never keep financial information on the Internet

Security, but cryptography optional

No special software

Work with today's Web, FTP, and E-Mail

High degree of privacy

Very low cost ($0.29 + 2%, INCLUSIVE)

Permit microtransactions (sub-penny level)

Available today!
(a working system, not a statement of direction)

II. Is Encryption the Solution?

Not completely. There are serious problems.

-- Legal: Export & patent restrictions.
(Do you like lawsuits & trials?)
(Do you want customers in France?)

-- Interoperation: There are NO standards.
(Do you want to do business TODAY?)
(Do you want to limit your customer base
to browser X? To only the WWW?)

-- Usability: Harder than setting your VCR clock.
(Do you want Joe Sixpack as a customer?)
(Do you want to be liable for mismanaged
crypto keys? Deleted wallet files?)

III. Why Information Commerce is Special

Information has unique properties

For Buyers
-- Can't evaluate without obtaining

For Sellers
-- Never need returned goods
-- Near-zero cost of inventory and distribution
-- Relatively low out-of-pocket expenses

What should a commerce mechanism guarantee?

Credit cards guarantee, for physical goods:
-- Merchant gets paid, OR gets back goods
-- Statistically, usually gets paid.

Information merchants NEVER need returns.
-- The statistical guarantee is everything!

Commerce and Inventory

Commerce evolved for goods or services.
-- Information was bundled (e.g., in a book)

Inventory was costly:
-- Shipping, handling, trucking, warehousing, etc.
-- Returned goods, damaged merchandise, etc.

The cost of producing information (e.g., royalties)
is small, compared to the cost of inventory

Information Networks Change the Rules

Electronic information can be
-- duplicated cheaply (virtually free)
-- distributred cheaply (virtually free)

Of course, Internet access does cost real money
-- Each subscriber pays for their own "pipe"

The dominant cost is now information
creation/synthesis, NOT the cost of inventory!

IV. The First Virtual Information Commerce Model

We Introduced a New Paradigm:

Duplicate, Download, and Decide to Pay

Consistent with the Internet culture:
-- Information is still freely available!
-- If you don't like what you get, you don't pay.
-- But if you abuse the system you get no more!

Allows anyone to be a seller or buyer,
-- No need to qualify as a Visa/MC merchant.
-- An unprecedented opportunity for small entrepreneurs.

Works with existing financial institutions
-- Keeps sensitive information off the Internet
-- Use bankcards/checking accounts for settlement

Benefits to Sellers

Accounting, billing, and collection are simplified
-- First Virtual serves as the "back office"

The entire Internet is enabled as the market
-- 30 million potential customers today!

"Shareware" is the proof of concept
-- Some folks have earned a lot of income.
-- Ease of payment has proven crucial.
(Consider the first shareware millionaire!)

The FV paradigm is vastly superior to shareware.
-- With FV, payment is automatic.
-- With FV, abusers lose privileges.
-- With FV, it takes action to refuse to pay.
-- FV works well for text, music,
images, & software.

Benefits to Buyers

Examine information before deciding to pay for it

E-mail confirmation of buyer acceptance detects fraud

Bankcard and checking information is not sent or stored on the Internet

No cryptography is needed!

No new software is needed!

For privacy, cryptography can be layered on optionally later.

V. How It Works

Account Application via Internet
-- Sensitive information via phone/post

Transactions via Web/FTP/Email
-- Payment confirmation via email

Anyone can be a seller
-- Those without servers can use our
InfoHaus (TM), an information
"automat"

Settlement via direct bank deposit
-- Payment lags for untrusted sellers

The Account Application Process

Buyer fills out application form via Telnet, Web, Email, etc.

Includes all non-sensitive information:
-- Name
-- Email address
-- Address, phone, etc.
-- Account-ID choice
-- Optional settings

If complete, FV sends "almost ready" message, including 12-digit application-id

To activate your account as a BUYER:
-- Call 1-800-*** with app-id and CC#
-- $2 charge on credit card
-- Credit card used for future purchases

To activate your account as a SELLER:
-- Send a $10 check to PO Box ****
-- Bank account credited for future sales

Notification by email when account is ready to use for buying, selling, or both.

The Transaction Process

Seller receives order via Web, FTP, E-Mail, etc.

Seller may validate account in real-time or
via e-mail

If seller honors order
-- Information sent to buyer
-- Transaction record sent to First Virtual

First Virtual sends "transfer token" to buyer
via e-mail

Buyer replies with 1 word:
-- Yes: payment is authorized.
-- No: payment is declined.
-- Fraud: cancel account.

Seller bears risk of non-payment,
but buyers who say "no" too often
get invalidated

The First Virtual InfoHaus

The World's First Public Access Information Mall

Anyone can be a seller!
-- Each seller has a seller description + info-items
-- Each info-item has a free part and a paid part
-- 7x24 high-bandwidth availability
-- Topic/keyword and date-based searching
-- Full text search soon

What can be sold?
-- One-shot items for sale
-- Periodicals (magazines)
-- Boxed sets
-- Web forms that generate charges, send email

Seller access methods
-- Telnet/FTP interface
-- MIME email interface

Buyer access methods
-- WWW
-- FTP
-- Email

The InfoHaus: An Automat for Cyberspace!

All operations automated
-- Seller does not need own computer
-- FV does backups, collection, etc

Reasonable charges for use
-- Monthly storage fees
-- Transaction fees for completed sales
(+ normal FV charges)

Accounting summaries generated (soon)
-- How many browsed
-- Who bought, who refused

A Third party business?
-- The InfoHaus has NO privileged position
-- Separate business unit
-- Competition with FV possible

The Settlement Process

When buyer accumulates more than $N of charges:
-- A single bankcard transaction is originated
-- Buyer gets a detailed summary via e-mail

First Virtual ages funds for 90 days (for untrusted sellers), and then regularly:
-- Direct-Deposits money into seller's account
-- Notifies seller in detail via e-mail

The 90 day period will eventually be reduced for some sellers

Chargebacks are STRONGLY discouraged.

VI. What It Costs

Account setup
-- For buyers: $2.00
-- For sellers: $10.00

Transaction fees
-- For buyers: None
-- For sellers: $0.29 + 2%
(Visa/MC/bank charges INCLUDED)
-- Microtransactions permitted
(seller accumulation required)

Seller settlement charge
-- $1.00 per settlement

InfoHaus Charges
-- $1.50/month/meg
-- 8% of sucessful sales
(+ normal FV charges, as above)

VII. The First Virtual Protocols

The First Virtual System Architecture

-- Client/server protocols based on TCP/IP standards
-- Client sends a "transaction" to FV server (Finer granularity than "fund transfer")
-- Server gives response, may take additional actions

Data Structures
-- All transactions/responses are structured MIME data
-- Heavy use of multipart/mixed, multipart/alternative
-- One new specialized MIME type

Client Interfaces
-- All access may be done via email (SMTP)
-- Optional access through interactive protocol (SMXP)
-- Specialized uses of finger, telnet, DNS, etc.

Connection Protocols: Mail and SMXP

Mail is simplest:

To: sgcs@card.com
Content-type: application/green-commerce;
transaction=transfer-request

SMXP is almost as easy

New protocol (port 440), modeled on POP protocol

Basic function: interactive exchange of MIME objects.

Client passes one MIME object to server.
-- Server gives back "+ERR" or "+OK"
-- Sometimes, server also gives back another MIME object.

The application/green-commerce MIME Type

One mandatory Content-type parameter:
-- "Transaction" tells what kind of action

Simple data structure:
-- Modelled on RFC 822
-- "Attribute: Value" syntax
-- RFC 822 parsers are instantly reusable

A Simple Example

To: sgcs@card.com
Content-type: application/green-commerce;
  transaction=transfer-request

BUYER: Joe Is a cool dude
SELLER: CrazyRDIM
AMOUNT: 19.99
CURRENCY: USD US Dollars
TRANSFER-ID: <CR42@somewhere.com>
SECURITY-REQUIREMENTS: None
DESCRIPTION: Purchase of used stock quotes

We gave you lots of useful stock quotes and tips that may have helped you in your investments, and it isn't our fault if a grand jury is interested in you!

Account Setup Transactions

APPLICATION-REQUEST:
-- User requests application:

APPLICATION-RESULT:
-- Server sends back form

NEWACCT-REQUEST:
-- User send back answers

NEWACCT-RESULT:
-- Server sends back result:
    May send more than one (pending, accepted)
    Sensitive information entered by phone/post

Fund transfer (purchase)

TRANSFER-REQUEST:
-- Someone requests payment

TRANSFER-QUERY:
-- Server asks buyer for confirmation via email

TRANSFER-RESPONSE:
-- User says yes/no/fraud
-- Special hack: SERVER-ID in Subject.

TRANSFER-RESULT:
-- Server tells initiator of result.

Other Transactions

Account status inquiries & histories
INQUIRY-REQUEST, INQUIRY-RESULT
HISTORY-REQUEST (accountholder only)

Server capability inquiries
CAPABILITIES-REQUEST,
CAPABILITIES-RESULT

Tracking the money flow:
PAYIN-NOTIFICATION
PAYOUT-NOTIFICATION
PAYIN-CHARGEBACK-NOTIFICATION
PAYOUT-CHARGEBACK-NOTIFICATION
PAYIN-FAILURE-NOTIFICATION
COLLECTION-FAILURE-NOTIFICATION

Changing account information:
INITCHG-REQUEST, INITCHG-RESULT,
CHGACCT-REQUEST, CHGACCT-QUERY,
CHGACCT-RESPONSE, CHGACCT-RESULT
(Changes to sensitive information via phone)
(Changes to email address routed through
old address for confirmation)

Miscellany

For testing: test.card.com

For mail to accountholders:
accountid@relay.card.com
(Normalize account id to alphanumerics)

For quick inquiry-request:
finger account-id@card.com

For brain-damaged gateways:

localpart%domain@email.challenged.card.com

VIII. The Corporation

Founders:
- Lee Stein, President and CEO
- Einar Stefferud, Chief Visionary
- Nathaniel Borenstein, Chief Scientist
- Marshall T. Rose, Principal

Technical Leaders:
- Carlyn Lowery, Operations Management
- Darren New, InfoHaus Development

Strategic Partners:
- EDS
- First USA
- NDMC
- Lloyd Internetworking
- NCD/Z-Code
- (several others in process)

First Virtual's and the Internet Culture

We wanted to preserve the Internet's spirit while enabling commerce.

-- Specifications published openly
    (on ftp.fv.com today)
    (soon-to-be Informational RFC's)
-- Approach leverages existing infrastructure
-- Freely-available software
-- International usability
-- Commercial support available
-- System operational prior to public announcement
    (NO VAPORWARE!)
-- System operated on a cost-recovery basis

IX. The Future

We're not standing still!
-- Second payment system under development
-- Additional technologies in planning stages
-- More major merchants coming soon
-- Watch for our own information products soon!

-- Our goal: to be the leaders in intellectual property commerce.

It Works Today! Here's How to Start:

General information:
-- info@fv.com
-- http://www.fv.com

Account Application:
-- apply@card.com
-- telnet telnet.card.com

For sellers:
To use the InfoHaus:
infohaus-guide@fv.com
Technical specs:
URL ftp://ftp.fv.com/pub/docs/*
Format is easy enough to type by hand!
Free code:
Patch kits for WU ftpd
CGI scripts for Web servers
Shell programs and C API for programmers
URL ftp://ftp.fv.com/pub/code/*
Commercial support: Lloyd Internetworking

For buyers:
No new software needed.
Browse the InfoHaus:
http://www.infohaus.com

X. What We've Learned From the first 8 months of Internet Commerce

What we've learned so far basically comes in four categories

-- Technical Stuff
-- Political Realities
-- Customer Service Requirements
-- The Nature of the Market

Technical Stuff:

Bottlenecks can lurk anywhere
-- We designed our system for huge volumes
-- Off-the-shelf stuff (e.g. web servers) caused unexpected bottlenecks

Exponential growth has to start somewhere
-- 15% a week is amazing once it gets going
-- It can take a couple of months to get going

Programmers are users too
-- Tools for commerce-enabling merchants must be extremely usable
-- The people who want to use these mechanisms are not all rocket scientists (or even computer scientists)

More Technical Stuff

Internet mail is a mess
-- There are zillions of out-of-spec implementations
-- We knew & designed for that fact
-- It was still worse than we expected
(and not really improving)
-- Mail-based services must be pro-active
and robust
-- Other protocols (HTTP, etc) are just as bad

Without automation you're dead
-- A one-time event in May 1995 is a
1000-time event in May 1996
-- Every anomaly must be automated
-- Users both cause and discover anomalies

Political Realities

There are powerful vested interests
-- Build a better mousetrap, and some may deny it
-- Technological excellence is only step one.

Underlying dynamic: collision between 2 worlds:
-- The Internet & The Financial World

We focused on Internet sensitivities
-- Openness (buyers & sellers)
-- Evolution
-- Compatibility with Infrastructure & Culture

We almost missed some financial world sensitivities
-- Openness (processors & banks)
-- Alliances with current players

Customer Service Requirements

-- Customers get confused.
-- Plan on ramping up.
-- Need explicit mechanism for ramping up (crisis response)
-- Need followup tracking & quality control.
-- Instructions can't be too clear or they won't be read.
-- Customer problems drive automation improvements.

Bottom line: under control, but with more supporting software & effort than we anticipated

The Nature of the Market

Why would anyone want to do commerce on the net?

Some early answers:
-- Internet-related information
-- Software & financial services
-- Erotica

FV's answer so far:
-- Nobody really knows yet
-- This is the hard part.
It's also where the real money is.

Advice to entrepreneurs:
-- Use any payment system that works
(so far just FV, but expect to use several)
-- Focus on figuring out what people will pay you for. Any payment mechanism that works will suffice.
-- Let people know what you're selling.
Few will just stumble in.

Payment Switches for Open Networks

David K. Gifford*†
Lawrence C. Stewart
Andrew C. Payne
G. Winfield Treese
Open Market, Inc. ‡

Abstract

We describe the first operational Internet payment switch that provides real-time authorization suitable for direct use by merchant servers. A payment switch is a server that creates digital representations of conventional financial instruments, and forwards authentic payment orders on these instruments to their corresponding conventional financial networks and institutions. Our payment switch provides support for time-based and item-based pricing, implements switch based authorization and settlement aggregation for micro-payments, and includes an extensive customer support system in order to provide a high level of customer confidence in electronic commerce. Fraud control is based on a transaction-specific multi-level security model that accommodates existing Internet browsers. Multiple authentication technologies are applied to every transaction.

1 Introduction

The recent rapid growth of information applications on the Internet suggests that public computer networks have the potential to establish a new kind of open marketplace for goods and services. A central requirement for a marketplace is a payment mechanism. However there is no merchant independent payment mechanism available for computer networks, that permits users to utilize conventional financial instruments such as credit cards, debit cards, and demand deposit account balances for real-time purchases. We expect that both retail payment and wholesale payment mechanisms will be required for networks; consumers will use the retail mechanism for modest size purchases, and institutions will use the wholesale mechanism for performing settlement between trading partners. In order to achieve wide acceptance the retail mechanism will need to be a logical evolution of existing credit-card [10], debit card [9], and Automated Clearing House facilities. Similarly the wholesale mechanism will need to be an evolved version of corporate electronic funds transfer [4].

An unavoidable property of public computer networks is that they are comprised of switching, transmission, and host computer components controlled by many individuals and organizations. Thus it is impossible for a network payment system to depend upon a specified minimum required degree of software, hardware, and physical security for all of the components in a public network. For example, secret keys stored in a given user's personal computer can be compromised, switches can be tampered with to redirect traffic, and transmission facilities can be intercepted and manipulated. We have developed threat models for open networks. These threat models have shown us that cryptographic protocols, while important, do not in and of themselves solve the security requirements of payment services. Our findings are consistent with Anderson's work on security failures in electronic banking systems [2].

The risk of performing retail payment in a public network is compounded by statutes that make a payment system operator in part liable for the security lapses of its users. Existing Federal statutes in the United States, including the Electronic Funds Transfer Act and the Consumer Credit Protection Act, re-

*Also with the Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA.

†The authors can be reached via electronic mail at {gifford,stewart,payne,treese}@openmarket.com.

quire the operator of a payment mechanism to limit consumer liability in many cases. Payment system operators may have other fiduciary responsibilities for wholesale transactions. Similar responsibilities exist in other countries for retail and wholesale transactions [3].

Conventional commerce systems are designed to allow merchants to conduct business without exposing the merchants to undue risk. For example, in existing credit card payment systems, a credit card's issuing bank takes on the fraud risk associated with misuse of the card when a merchant follows established card acceptance protocols. Standard acceptance protocols can include verifying a cardholder's signature as like the one on the back of the card, and obtaining on-line authorization. However, in network based commerce a merchant cannot physically examine a purchaser's credit card, and thus the fraud risk may revert to the merchant in so called "card not present" transactions. Many merchants cannot take this risk because of their limited financial resources.

A fundamental goal of our work is to create a network payment service that will reduce the risk of network fraud, and thus permit more merchants to participate in network commerce. A payment switch is a network service that authorizes and executes digital payment orders which are backed by external accounts. A payment switch authenticates a payment order, checks for sufficient funds or credit, and then originates funds transfer transactions to carry out the payment order. A payment switch acknowledges acceptance or rejection of a payment order. More than one payment switch may exist on a given network, and a given payment switch may operate on more than one host to increase the payment switch's reliability, availability, and performance.

In the remainder of this paper we will discuss related work (Section 2), our payment switch technology (Section 3), and close with observations on our operational experience and plans (Section 4).

2 Related Work

Our work builds on related work in network based order entry, on-line payment servers, and off-line digital cash:

Network based order entry. Many Internet merchants allow buyers to complete network based order forms for hard goods. Network based order entry systems perform merchandise ship time authentication of the credit card account used for payment. In most cases, order form transactions are unencrypted and thus are vulnerable to eavesdropping. Certain merchants permit users to set up accounts with associated credit card numbers. In these systems merchandise is charged to a user's account. This eliminates the need for credit card information to pass in the clear over the network each time an order is placed. In the near future it is likely that cryptographic protocols will provide communication security for order entry. In most cases, even with cryptographic protocols, common network threats such as workstation compromise, Trojan horse client software, and the use of stolen credit card instruments are not addressed.

On-line payment servers. Recent work on payment servers is closely related to our work, but existing payment servers do not connect to the financial system for authorization or do not provide unforgeable real-time authorizations suitable for direct use by merchant servers. Existing network payment system proposals include the Simple Network Payment Protocol [7], CMU's Internet Billing Server [13], and ISI's NetCash [12]. Though these payment servers do not require trust between the parties in a transaction, the parties must trust the payment server, and the payment server must be on-line during the transaction. Kristol, Low, and Maxemchuk [11] propose a system for anonymous payment that protects all parties from being cheated. The key feature in their system is the careful separation of information, so that only "need to know" information is exposed to any participant in the transaction.

Another on-line payment server is the First Virtual system. The First Virtual [8] payment server provides each user with an account with a unique identifier backed by a credit card. A customer gives the identifier to a vendor when purchasing an item. The vendor reports the transaction to First Virtual, which sends electronic mail to the customer asking for confirmation of the transaction. The customer may approve the transaction, refuse payment, or claim that fraud has occurred. The vendor is at some risk because the customer may refuse to pay. Because the marginal cost of information products is low, vendors of such products may often find this risk acceptable.

Off-line digital cash. It is possible that network payment systems will include support for digital cash. Digital cash is a negotiable financial instrument that is an analog of conventional cash. An advantage of digital cash is that users can exchange value between themselves without using an on-line server. In order to permit off-line operation, digital cash systems require protected devices, such as smart cards, that are

tamperproof [1]. If desired, digital cash transactions can be anonymous [6].

Our system has several benefits not found in many of these previous approaches. These include:

Digital analogs of conventional financial instruments. When an authentic payment order is presented to a payment switch, a corresponding conventional financial transaction is executed in real-time. Confirmation of the transaction is immediately available to relevant principals. Payment switch interfaces to external financial systems are implemented as modules that permit a payment switch to be easily expanded to handle new financial institutions or payment instruments.

The First Virtual payment system is the closest to our approach, but it does not provide real-time confirmation suitable for direct use by merchant servers because of the delay inherent in their electronic mail confirmation system. Thus, while the First Virtual system can be used for real-time vending of soft goods such as documents and software, the vendor takes on the risk of a valid account being unable to pay when the transaction is settled later.

Multiple authentication technologies. A payment switch applies more than one authentication technique to every request, and the techniques used depend upon the transaction type and value. Both system and user determined policies are applied in determining which authentication technologies are employed. This multilevel approach permits the balance between security and customer convenience to be adjusted depending on the risk profile of a transaction and a merchant. In addition, using multiple techniques guards against the complete failure of a single technique.

Extensive customer service support. A payment switch maintains an automated customer support system that provides information on committed payment transactions. A payment switch maintains complete logs of all transactions, and provides every buyer and merchant with Smart Statements[1] that describe their activity. Smart statements provide a fixed point of access for goods purchased, allowing a buyer to view transaction detail information and to fill out automated customer service forms on the products purchased. Smart statement entries that correspond to information products allow a buyer to view the product again directly from the smart statement. In the event of communication failure during the commit of a purchase transaction a buyer can refer to his or her smart statement to discover the transaction's disposition. The customer service support provided by a payment switch provides a high level of buyer comfort because buyers are assured of a complete accounting of their purchases as well as an effective means to resolve questions.

Compatibility with existing protocols. Our approach is based on existing network protocols, such as HTTP [5]. Thus existing browsers can be used without modification in many forms of electronic commerce using our payment switch and merchant server systems.

[1] Smart Statement is a trademark of Open Market, Inc.

3 The Payment System

The overall architecture of our payment system is a distributed collection of clients, content servers, and payment switches that are not under common administrative control. Client capacity, performance, and availability can be increased by adding network capacity, content servers, and payment switches.

We will introduce our payment system by following a typical payment transaction, and then return to discuss how the payment switch functions as part of this transaction and elaborations of the basic protocol. In our discussion we will use the following terms:

Buyer The principal purchasing an item.

Merchant The principal selling an item.

Client The software being used by a buyer.

Content Server The software being used by a merchant.

Authenticated The payment switch has decided that a transaction was originated by the buyer specified by the transaction within a degree of certainty appropriate for the transaction, the buyer, and the merchant.

Authorized The payment switch has decided a buyer has sufficient funds or credit for a given transaction.

Committed The payment switch has decided to settle a transaction, and funds will flow from the buyer to the merchant.

The ten messages involved in a typical purchase transaction (Figure 1) are:

1. The client requests a product description page from the content server using HTTP using a GET request. The page includes one or more payment URLs that describe products.

   A payment URL (Uniform Resource Locator) is a string that includes:

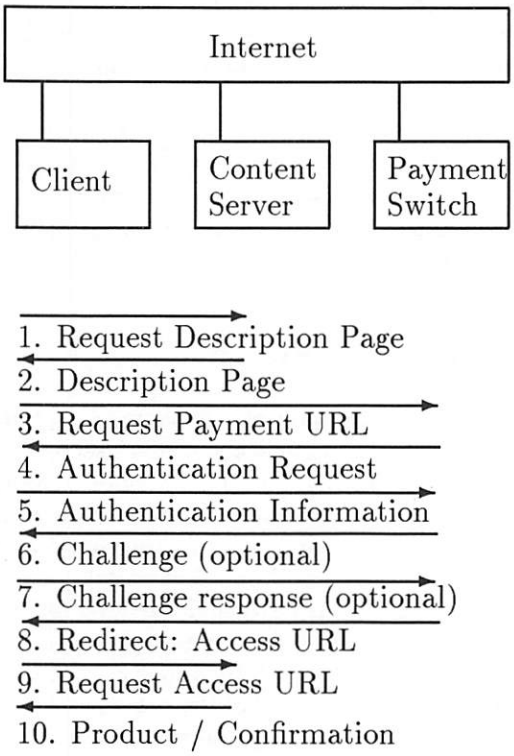


Figure 1: Purchase transaction steps

Payment switch host name The host the client will contact when this URL is activated.

Merchant identifier The principal identifier of the merchant selling the product. This is principal which will be paid when a sale is committed using the payment URL.

Product price The product price and currency. Multiple currencies are supported, including vendor currencies such as frequent flyer miles.

Target URL for product fulfillment This describes a URL that will provide product fulfillment. In the case of information products, this is a description of the information being sold. In the case of hard goods, this is a description of the product that will be fulfilled by the merchant server.

Expiration time of the payment URL A time and date at which the offer represented by the payment URL will expire.

Authenticator A digital signature of the payment URL using the private key of the merchant.

2. The product description page is returned to the client and displayed. The client then activates one of the payment URLs to purchase the product the URL describes.

3. The payment URL is forwarded to the payment switch by the client.

4. The payment switch requests the principal identifier of the buyer and the principal's password.

5. The principal identifier and password are sent to the payment switch. The payment switch now knows the sending principal (the buyer), the beneficiary principal (the merchant), the amount of the transaction, the product being purchased, and the URL to be used for product fulfillment. The payment switch also knows the sender's apparent IP address, the client type and its capabilities, and the type of channel (insecure, secure) that the client is using. Based on all of this information, the payment switch chooses a set of authentication techniques appropriate to the buyer, the merchant, and the transaction.

6. (Optional) If further authentication is required, a challenge is presented to the client. This challenge is based upon parameters previously set by the switch and the buying principal. Challenges can include client specific challenges, buyer specific challenges such as secondary passwords, and challenges to a buyer's smart card.

7. (Optional) The challenge response is returned to the payment switch. The payment switch can now authenticate the buyer based on the information the switch has received. If the payment switch decides that the request is not authentic, a rejection page is returned. If the payment switch decides the request is authentic, then the switch proceeds to authorization.

   Authorization is performed with respect to the financial instrument that backs a buying principal's account, and includes a switch specific component and an external financial system component. Switch based authorization includes negative and positive file checks, principal set spending limits and allowable product codes, address checks, velocity checks, and account type specific checks. Assuming that a transaction passes all switch based authorization checks, it is forwarded

to the external financial institution that is responsible for the payment instrument connected with the buyer's account.

The payment switch is designed to handle "micropayments" even at the sub-cent level. Certain classes of authorizations for the same financial instrument are aggregated at the payment switch to reduce the load on external financial networks and to improve system performance. When authorizations are aggregated the payment switch may authorize a transaction without consulting an external financial system.

If a transaction is not authorized, a rejection page is returned to the buyer.

If a transaction is authorized, then it is recorded in a settlement log. Once a transaction has been stably recorded in the settlement log it is said to be committed, and settlement will occur.

8. Once a transaction is committed, an access URL is returned to the client as an HTTP redirect response. An access URL is a capability that permits a client to access the product which has just been purchased. An access URL includes:

   Content Server Name and Product Identifier The name of the server and product that were derived from the target URL in the payment URL.

   Buyer Authenticator A value that allows the content server to authenticate the buyer. At present, the buyer's IP address is included to restrict the access URL to delivering information only to the specified IP address. Other possibilities for buyer authenticators include the public key of the buyer, and legal ship to addresses.

   Expiration A time and date at which the access URL will expire. Fixed time access URLs are used to implement subscriptions.

   Access URL Authenticator A digital signature of the entire access URL that allows a content server to verify that the access URL was created by a payment switch.

9. The client forwards the access URL to the content server it describes. The content server validates the access URL, and either returns the product (soft goods), or queues a fulfillment order (hard goods).

10. Either the product or a description of the delivery instructions are delivered back to the client.

This basic processing flow is extended in our payment switch by other features designed to provide additional customer convenience.

For example, a buyer might like to shop for items and simultaneously a running total without committing to a purchase. A payment switch supports this functionality by using payment URLs called shopping cart URLs which do not cause a purchase to happen immediately. Instead, a product represented by a shopping cart URL is added to a buyer specific "shopping cart" at the payment switch when the shopping cart URL is activated by a buyer. By using a distinguished URL at the payment switch, a buyer can view his or her shopping cart, modify the contents of the cart, and atomically purchase all of the items in the cart.

The payment switch also provides buyers and merchants with direct access to a list of their respective committed transactions. In the case of buyer "smart statements," the statements include access URLs that will return the buyer to the purchased product. In addition, smart statements include extensive customer feedback capabilities.

The payment switch also implements duplicate purchase detection and principal specific account control. For example, a principal can set additional authorization criteria that limit an account to certain transaction types or dollar volumes. In addition, facilities for subscription and discount purchasing are included.

A payment switch includes modular authentication and settlement components which allow the switch to be readily adapted to new authentication protocols and additional financial networks and institutions (Figure 2). The form of payment orders can also be easily extended to accommodate new applications. As shown in Figure 2, the stable state of a payment switch is stored in a relational database. Roll-forward recovery is used to ensure that all transactions are properly processed in the face of payment switch and external financial system failures. Multiple transactions are used to process one payment order with intermediate state kept in the relational database. We have created a modular library of authentication techniques, and have written settlement modules for multiple financial networks in response to merchant requirements.

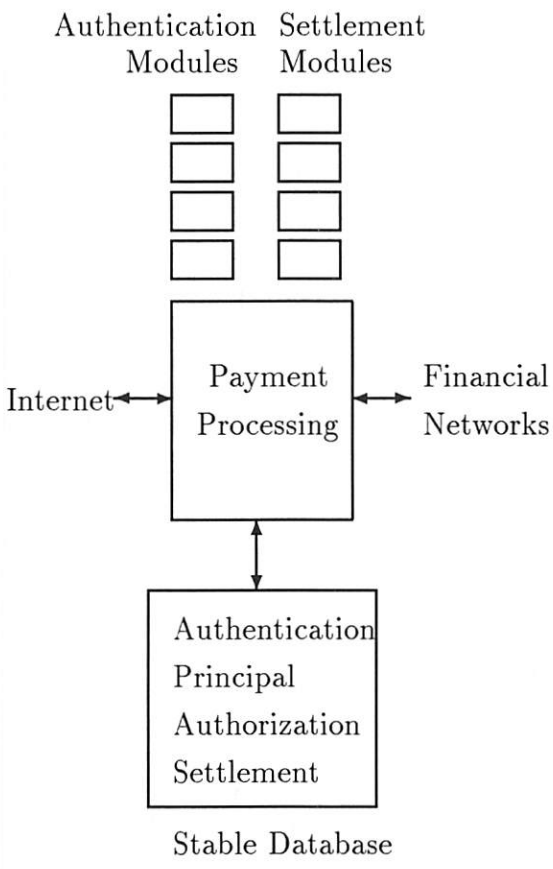


Figure 2: Payment switch internals

4 Conclusion

Payment switches that interconnect open networks with conventional financial instruments promise to permit new kinds of commerce. Our first payment switch went on-line in October 1994 and has been used continuously since then by Internet users. We have discovered that our provision for high quality automated customer service is an important component of our payment services. We are planning to make available new payment protocols in addition to our HTTP based payment service in the next year. We are also expanding our payment switch to run on multiple hosts to handle transaction volume growth and to permit merchants to operate their own payment switches.

It is our belief that electronic commerce offers a new frontier for both retail merchants and wholesale trading partners. The possibilities for doing business in an electronic medium seem limitless, and should allow new business models to flourish that will benefit the world.

References

[1] Anderson, R., UEPS–A Second Generation Electronic Wallet, Proc. of the Second European Symposium and Research in Computer Security (ESORICS), Touluse, France, November, 1992.

[2] Anderson, R., Why Cryptosystems Fail, Proc. 1st Conf. Computer and Comm. Security '93, November 1993, Virginia, Assoc. Computing Machinery, pp. 215-227.

[3] Bank Administration Institute, Payment Systems in Eleven Developed Countries, 1989.

[4] Bender, M., EFTS: Electronic Funds Transfer Systems, Kennikat Press, Port Washington, New York, 1975, pp. 43-46.

[5] Berners-Lee, T., Fielding R., Frystyk Neilsen, H., Hypertext Transfer Protocol–HTTP/1.0, IETF Internet Draft, December, 1994.

[6] Chaum, D., Achieving Electronic Privacy, Scientific American, August 1992, pp. 96-101.

[7] Dukach, S., SNPP: A Simple Network Payment Protocol, MIT Laboratory for Computer Science, Cambridge, MA, 1993.

[8] First Virtual, Inc., Introducing the First Virtual Internet Payment System for Information Commerce, 1994. Also see http://www.fv.com.

[9] Gifford, D., and Spector, S., Case Study: The CIRRUS Banking Network, Comm. ACM 8, 28 (August 1985), pp. 797-808.

[10] International Standards Orgnaization, ISO 8583: Bank card originated messages–interchange message specifications–content for financial transactions, August, 1987.

[11] Kristol, D., Low, S., and Maxemchuk, N., Anonymous Internet Mercantile Protocol, AT&T Bell Laboratories, March, 1994.

[12] Medvinsy, G., and Neuman, B. C., NetCash: A Design for Practical Electronic Currency on the Internet, Proc. 1st ACM Conf. on Comp. and Comm. Security, November, 1993.

[13] Sirbu, M. A., Internet Billing Service Design and Prototype Implementation, Information Networking Program, Carnegie Mellon University, 1993.

NetBill Security and Transaction Protocol

Benjamin Cox
thoth+@cmu.edu

J. D. Tygar
tygar@cmu.edu

Marvin Sirbu
sirbu+@cmu.edu

Carnegie Mellon University
Pittsburgh, PA 15213–3890

Abstract

NetBill is a system for micropayments for information goods on the Internet. This paper presents the NetBill protocol and describes its security and transactional features. Among our key innovations are:

- *An atomic certified delivery method so that a customer pays if and only if she receives her information goods intact.*
- *Outsourcing access control: different users can use different access control servers.*
- *A credential mechanism allowing users to prove membership in groups. This supports discounts.*
- *A structure for constructing pseudonyms to protect the identities of consumers.*

1. Introduction and Overview

Buyers and sellers increasingly want to use the Internet to conduct their business electronically. As a base for commerce, the Internet poses special challenges due to its lack of standard security mechanisms. At the same time, the ease with which buyers can peruse catalogs published via the World Wide Web makes the Internet attractive for commerce. Consumers will want to use the Internet as a means for multiple phases of the purchase process: searching for suppliers, price negotiation, ordering, and payment for goods. In the case of information items, such as software or journal pages, the Internet can deliver the items.

Using the Internet for commerce poses new variations on traditional issues. Transactions occur in cyberspace with no easily identifiable place of business for the merchant or physical delivery site for the customer. Transactions are subject to observation by third parties sharing the network. And the use of computers to support transactions makes record keeping easier, exacerbating privacy problems arising from transaction data collection by merchants.

Supporting transactions in cyberspace requires electronic analogs for many familiar procedures from face-to-face transactions. Parties need to know with whom they are dealing, or at least verify their creditworthiness. They need to be able to negotiate prices, perhaps providing credentials entitling them to special discounts, such as a student ID. Parents need methods to control where their children shop in cyberspace. In the case of information goods, the value of an item may be as low as a few cents, requiring transaction mechanisms which impose per-transaction overheads much smaller than those for typical check and credit card purchases. Merchants need to restrict the class of customers they support based on their credentials, to restrict distribution of sensitive materials.

We are building a system called NetBill which is optimized for the selling and delivery of low-priced network goods. A customer, represented by a client computer, wishes to buy information from a merchant's server. An account server (the NetBill server), maintains accounts for both customers and merchants, linked to conventional financial institutions. A NetBill transaction transfers information goods from merchant to customer, debiting the customer's NetBill account and crediting the merchant's account for the value of the goods. When necessary, funds in a customer's NetBill account can be replenished from a bank or credit card; similarly, funds in a merchant's NetBill account are made available by depositing them in the merchant's bank account. NetBill acts as an aggregator to combine many small transactions into larger conventional transactions, amortizing conventional overhead fees.

The transfer of an information good consists of delivering bits to the customer. Users may be charged on a per item basis, by a subscription allowing unlimited

Sponsored by the Air Force Materiel Command, under Advanced Research Projects Agency contract No. F19628–95–C–0018, "Electronic Commerce: The NetBill Project." Additional support was received from the National Science Foundation under Cooperative Agreement No. IRI–9411299, and from Visa International.
The views and conclusion contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the National Science Foundation, or the U.S. Government.

access, or by a number of other pricing models. A more detailed examination of the NetBill model may be found in [10].

NetBill requires an efficient set of protocols to support price negotiation, goods delivery and payment. This paper outlines our protocols.

Among our key innovations are:

- A method of (atomic) certified delivery so that a customer pays if and only if she receives her information goods intact.
- A system for allowing access control to be outsourced—so that different users may use different access control servers. (For example, some children's accesses may be governed by a PTA access control server, while other children may be under the domain of access control servers set up by a church group.)
- A credential mechanism for allowing users to easily prove membership in groups, to qualify for discounts or for other purposes.
- A structure for easily constructing pseudonyms so that buyers of information can protect their identities.

2. The NetBill Transaction Model

The NetBill transaction model involves three parties: the customer, the merchant and the NetBill transaction server. A transaction involves three phases: price negotiation, goods delivery, and payment. For information goods which can be delivered over the network, the NetBill protocol links goods delivery and payment into a single atomic transaction.

In a NetBill transaction, the customer and merchant interact with each other in the first two phases; the NetBill server is not involved until the payment phase, when the merchant submits a transaction request. The customer contacts the NetBill server directly only in the case of communications failure or when requesting administrative functions. Figure 1 shows the relationships among parties in a NetBill transaction.

2.1. Transaction Objectives

For a NetBill transaction, we have the following set of objectives. (Similar versions of objectives (a)–(d) below can be found in [2].)

a) Only authorized customers can charge against a NetBill account.

b) The customer and merchant must agree on the item to be purchased and the price to be charged.

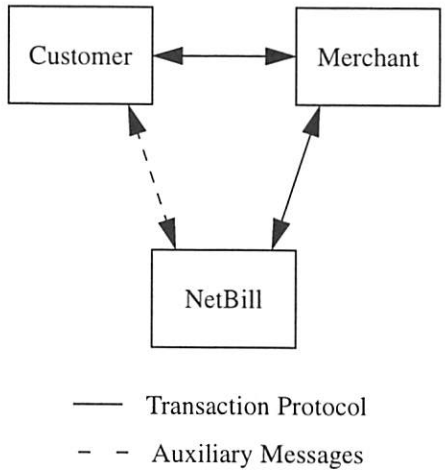


Figure 1: Parties in a NetBill Transaction.

c) A customer can optionally protect her identity from merchants.

d) Customers and merchants are provided with proof of transaction results from NetBill.

In addition, we have the following objectives to support price negotiation and goods delivery.

e) There is an offer and acceptance negotiation phase between customer and merchant.

f) A customer may present credentials identifying her as entitled to special pricing or treatment.

g) A customer receives the information goods she purchases if and only if she is charged (and thus the merchant is paid) for the goods.

h) A customer may need approval from a fourth (access control) party before the NetBill server will allow a transaction.

Finally, we add as a general objective for all phases of the purchase process:

i) The privacy and integrity of communications is protected from observation or alteration by external parties.

To achieve these goals, the NetBill protocol provides for strong authentication and privacy, atomic payment and delivery protocols, and a flexible access control system.

In the price negotiation phase, the customer presents evidence of her identity, and (optionally) supplemental credentials, and requests a price quote on an item. The customer may also include a bid for the item. The merchant responds with a price offer.

In the second phase, the customer accepts or declines the offer. In the case of information goods, acceptance constitutes an order for network delivery. The merchant provisionally delivers the goods, under encryption, but withholds the key.

Key delivery is linked to completion of the third phase, the payment protocol. In this phase, the customer constructs, and digitally signs, an electronic payment order (or *EPO*) and sends it to the merchant. The merchant appends the key to the EPO and *endorses* (digitally signs) the EPO, forwarding it to the NetBill server. The NetBill server returns a digitally signed receipt, which includes the key, to the merchant, who forwards a copy to the customer.

3. The Transaction Protocol

We use the notation "X $\Longrightarrow$ Y" to indicate that *X* sends the specified message to *Y*. The basic protocol consists of eight steps, which are:

1. C $\Longrightarrow$ M Price request
2. M $\Longrightarrow$ C Price quote
3. C $\Longrightarrow$ M Goods request
4. M $\Longrightarrow$ C Goods, encrypted with a key *K*
5. C $\Longrightarrow$ M Signed Electronic Payment Order
6. M $\Longrightarrow$ N Endorsed EPO (including *K*)
7. N $\Longrightarrow$ M Signed result (including *K*)
8. M $\Longrightarrow$ C Signed result (including *K*)

Objective (a) from Section 2.1 is realized because the customer must be authenticated to NetBill before the EPO (generated in step 5) will be accepted (in step 6).

Objective (b) is achieved because the relevant information is included in the EPO which must be signed by the customer in step 5 and endorsed by the merchant in step 6.

Section 4.2 presents a mechanism implementing objective (c).

Objective (d) is realized through the digitally signed receipt from NetBill in step 7.

Objective (e) is achieved by the exchange in steps 1 and 2 of the protocol.

Section 5.1 presents a solution implementing objective (f).

Objective (g) is realized by the exchange in steps 4–8, which we call *certified delivery*. The customer first gets a version of the goods encrypted with a key *K*. The goods are also cryptographically checksummed. In this way, the customer uses the checksum to confirm that she received the goods without transmission error. The customer returns the checksum to the merchant together with other information, and that message is forwarded to the NetBill server. The key *K* that is needed to decrypt the goods is registered with the NetBill server and also sent to the customer (step 8). The exchange in steps 4–8 provides protection to the customer against fraud by the merchant. For example, suppose there is a discrepancy between what the customer ordered and what the merchant delivered. The customer can easily demonstrate the discrepancy to a third party (such as a judge). The customer has NetBill's receipt (step 7, forwarded in step 8) indicating what was ordered, the amount charged and the key *K* reported to NetBill by the merchant. The customer also has registered with NetBill the checksum of the encrypted goods. Thus if the goods are faulty (*e.g.*, purchased software doesn't run), it is easy to demonstrate that the problem lies with the goods as sent and not with any subsequent alteration. This certified delivery technique also protects the merchant by requiring the customer to pay and the payment to clear through the NetBill server before the customer gets the use of the goods. (A more general discussion of the role of certified delivery and atomicity in general for electronic commerce can be found in [13].)

Section 5.2 presents a solution for objective (h).

Objective (i) is realized by encrypting communications between all pairs of parties and providing integrity checks on those messages.

3.1. Notation

We use the following notation to denote cryptographic operations. *X* and *Y* always represent communicating parties. *K* always represents a cipher key. The protocol is a sequence of messages exchanged among three parties: *C*, the customer; *M*, the merchant; and *N*, the NetBill server.

T_{XY}(Identity)	A Kerberos ticket proving to *Y* that *X* is named by *Identity*, and establishing a session key *XY* shared between them.
CC(*Message*)	A cryptographic checksum of *Message*, using an algorithm such as the Secure Hash Algorithm (SHA) hash function presented in [5].

$E_K(Message)$	*Message*, encrypted by a symmetric cipher using key *K*. The key *K* may be denoted as *XY*, meaning that it is known only to parties *X* and *Y*. The encrypted message implicitly includes a nonce.
$E_{X\text{-}PUB}(Message)$	*Message*, encrypted in party *X*'s public key using the RSA cryptosystem as presented in [8].
$E_{X\text{-}PRI}(Message)$	*Message*, encrypted in party *Y*'s private key using RSA.
$[Message]_X$	*Message*, clearsigned by *X* using RSA public key cryptography. Clearsigning is implemented by forming *Message*, *Timestamp*, $E_{X\text{-}PRI}(CC(Message, Timestamp))$. This is computationally efficient and allows any party to read the *Message* text without needing *X*'s public key. The clearsigned item implicitly includes a nonce.
$[Message]_{X\text{-}DSA}$	*Message*, signed and timestamped by *X* using the Digital Signature Algorithm (DSA) as described in [6].
$\{Message\}^X$	*Message*, encrypted for *X* using RSA public key cryptography. For computational efficiency, this is implemented by forming $E_K(Message)$, $E_{X\text{-}PUB}(K)$. The encrypted message implicitly includes a nonce.

3.2. The Price Request Phase

This section assumes possession of tickets; the method for obtaining tickets is shown in Section 4. The *Identity* item may actually be a pseudonym, as shown in Section 4.2.

1.	$C \Longrightarrow M$	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2.	$M \Longrightarrow C$	E_{CM}(ProductID, Price, RequestFlags, TID)

These two steps represent a request/response message pair in which the customer requests a price quote of the merchant. The customer presents an identifying ticket (the identity presented may be a pseudonym; see Section 4.2 for details on pseudonyms) to the merchant, along with some optional *credentials* establishing her membership in groups which may make her eligible for a discount. (We discuss these credentials in Section 5.1.)

The customer passes parameters indicating Product Request Data (*PRD*, an arbitrary stream of application-specific data which the customer and merchant use to specify the goods) and some flags. These *RequestFlags* are the customer's indication of her request for the disposition of the transaction (*i.e.*, delivery instructions; see Section 3.6 for different transaction options).

The customer may also optionally include a Bid, indicating to the merchant a price she may be willing to pay for the item.

The Transaction ID, *TID*, is optional in step 1. Steps 1 and 2 may be repeated as needed until the customer and merchant can agree on a price; the TID is present to indicate to the merchant that this is a repeated request.

The merchant stores the *PRD* for later use in delivering the goods, generates a new set of *RequestFlags* based on its response to the customer's *RequestFlags*, and generates a price quote and a TID (if one was not supplied in step 1) to identify this transaction in later steps. The TID is not globally unique; it is used only by the customer and merchant to maintain context between them.

The *ProductID* returned by the merchant in step 2 is a human-readable product description that will appear on the customer's NetBill statement.

3.3. The Goods Delivery Phase

Once the customer and merchant have negotiated a price for the goods in question, the customer directs the merchant to deliver the goods by supplying the TID that was used in the price request phase:

3.	$C \Longrightarrow M$	T_{CM}(Identity), E_{CM}(TID)
4.	$M \Longrightarrow C$	E_K(Goods), $E_{CM}(CC(E_K(Goods)), EPOID)$

The merchant generates a unique symmetric cipher key *K*, encrypts the goods using this key and sends the encrypted goods to the customer, along with a cryptographic checksum computed on the encrypted goods, so that the customer will immediately detect any discrepancy before proceeding. The merchant also sends

an Electronic Payment Order ID, or *EPOID*, with the goods.

The EPOID is a globally unique identifier which will be used in the NetBill server's database to uniquely identify this transaction. It consists of three fields: a field identifying the merchant, a timestamp marking the time at the end of goods delivery, and a serial number to guarantee uniqueness.

The specification that the EPOID must be globally unique is used to prevent replay attacks, in which unscrupulous merchants reuse customers' old signed payment instructions. The timestamp portion of the EPOID is used to expire stale transactions; it must be generated at the end of goods delivery because the delivery (especially for very large goods) may take longer than the payment expiration time.

Because the goods are delivered encrypted in step 4, the customer cannot use them. The key *K* needed to decrypt the goods will be delivered in the payment phase, which follows.

3.4. The Payment Phase

After the encrypted goods are delivered, the customer submits payment to the merchant in the form of a signed Electronic Payment Order, or *EPO*. At any time before the signed EPO is submitted, a customer may abort the transaction and be in no danger of its being completed against her will. The submission of the signed EPO marks the "point of no return" for the customer.

An EPO consists of two sections, a clear part containing transaction information which is readable by the merchant and the NetBill server, and an encrypted part containing payment instructions which is readable only by the NetBill server. The clear part of the EPO includes:

- The customer's (possibly pseudonymous) identity
- The human-readable Product ID (from step 2)
- The negotiated price (from step 2)
- The merchant's identity
- The cryptographic checksum of the encrypted goods, to forestall debate over the content of the goods or whether they were received completely and correctly
- The cryptographic checksum of the Product Request Data (from step 1), to forestall debate over the details of the order
- The cryptographic checksum of the customer's account number with an account verification nonce, so that the merchant may verify that any supplied credentials (see Section 5.1) were used correctly
- The globally-unique EPOID

The encrypted part of the EPO includes:

- A ticket proving the customer's true identity
- Any required authorization tokens (see Section 5.2)
- The customer's NetBill account number
- The account verification nonce
- A customer memo field

The EPO is a tuple:

Identity,
ProductID,
Price,
M,
$CC(E_K(Goods))$,
CC(PRD),
CC(CAcct, AcctVN),
EPOID,
$T_{CN}(TrueIdentity)$,
E_{CN}(Authorization,
    CAcct,
    AcctVN,
    CMemo)

Henceforth, we use the terminology *EPO* to denote this tuple.

After the customer presents the signed EPO to the merchant, the merchant endorses it and forwards the endorsed EPO to the NetBill server. The endorsed EPO adds the merchant's account number, the merchant's memo field, and the goods decryption key, as well as the merchant's signature:

$[[EPO]_C, MAcct, MMemo, K]_M$

At any time before the endorsed EPO is submitted to the NetBill server, the merchant may abort the transaction and be in no danger of its being completed against his will. The submission of the endorsed EPO marks the "point of no return" for the merchant.

The phase containing the submission and endorsement of the EPO is denoted:

5. $C \Longrightarrow M$ $T_{CM}(Identity), E_{CM}([EPO]_C)$
6. $M \Longrightarrow N$ $T_{MN}(M), E_{MN}([[EPO]_C, MAcct, MMemo, K]_M)$

Upon receipt of the signed and endorsed EPO, the NetBill server makes a decision about the transaction and returns the result to the merchant, who in turn forwards it to the customer.

The NetBill server makes its decision based on verification of the signatures, the privileges of the users involved, the customer's account balance, and the uniqueness and freshness of the EPOID. It then issues a receipt containing the result code, the identities of the parties, the price and description of the goods, the EPOID, and the key *K* needed to decrypt the goods. The receipt is digitally signed by the NetBill server, using the Digital Signature Algorithm (DSA). The receipt is denoted:

Result, Identity, Price, ProductID, M, K, EPOID

For this step, DSA is used rather than RSA because of its relative performance. While RSA signatures may be verified quickly, they are time-consuming to create; DSA signatures, on the other hand, may be created quickly. By using RSA for customer and merchant signatures and DSA for NetBill signatures, we may shift some computing load away from the NetBill server.

Some of the resulting burden on the merchant can be lifted by recognizing that, from a business risk perspective, it may be sufficient for a merchant to verify only a random sample of receipts signed by the NetBill server. Since integrity and authenticity are assured by the symmetric key encryption protocol, only accountability is at stake.

This receipt is returned to the merchant, along with an indication of the customer's new account balance (encrypted so that only she may read it). The EPO ID is repeated in the customer-specific data to ensure that the merchant cannot replay data from an earlier transaction.

7. $N \Longrightarrow M$ $E_{MN}([Receipt]_{N\text{-}DSA}, E_{CN}(EPOID, CAcct, Bal, Flags))$

The *Flags* included in the customer-specific data indicate simple messages from the NetBill server to the customer; for example, that the account balance has reached a "low-water mark" and should be replenished soon.

8. $M \Longrightarrow C$ $E_{CM}([Receipt]_{N\text{-}DSA}, E_{CN}(EPOID, CAcct, Bal, Flags))$

In step 8, the merchant responds to the request from the customer in step 5, forwarding the messages returned by the NetBill server in step 7.

3.5. Status Query Exchange

In the event of communications failure after step 5 of the protocol, the customer or merchant may be unaware of the transaction's status. (Before step 5, the transaction may be aborted with no difficulty, as no parties have yet signaled their commitment.) The system supports a status query exchange for delivery of this information.

The request and response proceed as one of the following exchanges, assuming the information is available. In each case, an alternate response is possible, indicating that the queried party does not have the requested information (possibly indicating why).

- The merchant requests the transaction status from the NetBill server:

1. $M \Longrightarrow N$ $T_{MN}(M), E_{MN}(EPOID)$
2. $N \Longrightarrow M$ $E_{MN}([Receipt]_{N\text{-}DSA}, E_{CN}(EPOID, CAcct, Bal, Flags))$

- The customer requests the transaction status from the merchant:

1. $C \Longrightarrow M$ $T_{CM}(Identity), E_{CM}(EPOID)$
2. $M \Longrightarrow C$ $E_{CM}([Receipt]_{N\text{-}DSA}, E_{CN}(EPOID, CAcct, Bal, Flags))$

- The customer requests the transaction status from the NetBill server:

1. $C \Longrightarrow N$ $T_{CN}(TrueIdentity), E_{CN}(EPOID)$
2. $N \Longrightarrow C$ $E_{CN}([Receipt]_{N\text{-}DSA}, EPOID, CAcct, Bal, Flags)$

- The customer requests the transaction status from the merchant for a Non-NetBill transaction (see Section 3.6):

1. $C \Longrightarrow M$ $T_{CM}(Identity), E_{CM}(EPOID)$
2. $M \Longrightarrow C$ $E_{CM}(Result, K)$

3.6. Handling Zero-Priced Goods

We anticipate that many NetBill transactions will be for *subscription goods*; i.e., goods for which the customer's marginal price is zero. With zero-priced goods, fraud is less important, and so we make several refinements to make our protocol more efficient in these cases.

First, we include a flag in the *RequestFlags* field of the price request (step 1) informing the merchant "If the price for this product is zero, treat this message as an automatic request for the goods."

Zero-priced transactions do not need to go through the NetBill server, as long as both parties agree. We can put another flag in the *RequestFlags* that informs the merchant "I require a NetBill receipt for this transaction." If this flag is present, the merchant must submit the transaction to the NetBill server, even if the price is zero. (The merchant may also decide to submit a zero-price transaction to the NetBill server.)

A customer or merchant may want to use the NetBill server on a zero-priced transaction if they require a signed receipt from a third party confirming the transaction. While subscription goods may not require a receipt, a merchant may decide to put a zero-priced purchase through NetBill in a "buy ten, get one free" situation so as to be able to prove that he actually provided the free item.

The merchant may change his price quote depending on this flag; if NetBill charges the merchant for billing services, the merchant may want to recover this cost if the customer requests a NetBill receipt for what might otherwise be a zero-priced transaction.

Combinations of these flags allow us to support four basic types of zero-price delivery:

3.6.1. Type I: Zero-Price Certified Delivery

This protocol eliminates the separate product request phase. Because steps 2 and 4 from the original protocol are combined, we indicate that by making steps 2 and 4 into a single step labeled 2/4.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID), E_K(Goods), E_{CM}(CC(E_K(Goods)), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}([EPO]$_C$)
6.	M $\Rightarrow$ N	T_{MN}(M), E_{MN}([[EPO]$_C$, MAcct, MMemo, K]$_M$)
7.	N $\Rightarrow$ M	E_{MN}([Receipt]$_{N\text{-}DSA}$, E_{CN}(EPOID, CAcct, Bal, Flags))
8.	M $\Rightarrow$ C	E_{CM}([Receipt]$_{N\text{-}DSA}$, E_{CN}(EPOID, CAcct, Bal, Flags))

3.6.2. Type II: Certified Delivery without NetBill Server

This protocol improves on Type I by further eliminating the call to the NetBill server. With this modification, the payment phase becomes little more than an acknowledgment.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID), E_K(Goods), ECM(CC(E_K(Goods)), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(EPOID, CC(E_K(Goods)))
8.	M $\Rightarrow$ C	E_{CM}(Result, K)

3.6.3. Type III: Verified Delivery

This protocol is nearly the same as the Type II protocol, except that the goods are encrypted in the shared session key *CM*. This bypasses the certified delivery mechanism, allowing the customer's software to begin streaming the goods to a viewer rather than being obliged to wait until all the goods have been delivered before receiving the key.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID, Goods, CC(Goods), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(EPOID, CC(Goods))
8.	M $\Rightarrow$ C	E_{CM}(Result)

3.6.4. Type IV: Unverified Delivery

This protocol improves on Type III by eliminating the acknowledgment of goods delivery in the payment phase if the merchant does not need it.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID, Goods, CC(Goods))

4. Identities and Authentication

When a customer creates a NetBill account, she receives a unique User ID and generates the RSA public key pair

associated with that User ID. This key pair is certified by NetBill, and is used for signatures and authentication within the system. (See [4] for a discussion of public key certification.)

Section 3.4 showed how these signatures will be used in the payment phase of the protocol. However, our protocol also uses Kerberos tickets. The NetBill transaction protocol involves several phases, for price negotiation, goods delivery, and payment; only the last of these phases requires nonrepudiable signatures. Instead of using public key cryptography for message authentication and encryption throughout the NetBill system, we use symmetric cryptography because it offers significant performance advantages.

We use the public key cryptography infrastructure to alleviate problems with traditional symmetric-key Kerberos (see [12]):

Kerberos uses a two-level ticket scheme; to authenticate oneself to a Kerberos service, one must obtain a *service ticket*, which establishes a shared symmetric session key between the client and server, and establishes that the Kerberos Ticket Granting Server believes the client's identity. To obtain a service ticket, a client must first obtain a *ticket-granting ticket* (or TGT), which proves the client's identity to the Ticket Granting Server. A client obtains a TGT via request from a Key Distribution Center, or KDC.

The Kerberos KDC/TGT arrangement introduces two significant problems that we may alleviate using public key cryptography. First, because it maintains a shared symmetric cipher key with every principal in the system, it is an attractive target for attack; recovering from compromise of the KDC requires establishing new shared keys with all users of the system. Second, a KDC and TGT will be a communications or processing bottleneck if a large number of users present a heavy traffic load.

To eliminate the Ticket Granting Server, we replace the TGT with a public key certificate, allowing each service to act as its own Ticket Granting Server. That is, a user presents a service ticket request encrypted with a certified public key (we call this a *Public Key-based TGT*, or *PKTGT*), and receives in response a symmetric-cipher-based service ticket. This service ticket is identical in form to a Kerberos service ticket. The Key Distribution Center is replaced by a key repository. The protocol for a customer to obtain a service ticket for a merchant *M* is as follows (before this step occurs, the customer requests the merchant's public key certificate over any available channel—such as an unsecured remote procedure call):

1. $C \Longrightarrow M \quad [\{\text{Identity, M, Timestamp, K}\}^{M}]_{C}$
2. $M \Longrightarrow C \quad E_{K}(T_{CM}(\text{Identity}), CM)$

This model preserves the efficiency of symmetric ciphers for most communication and repeated authentication, and isolates the computational expense of public key cryptography to initial authentication between parties. We refer to this model as "Public Key Kerberos," or "PK Kerberos."

In the NetBill system, a customer obtains Kerberos tickets for the NetBill transaction server at the beginning of a session and obtains Kerberos tickets for merchants as she needs them. Merchant servers will continually maintain their own tickets for the NetBill transaction server.

4.1. Key Repository

Private keys are large, so users cannot be expected to remember them. Permanently storing private keys at a user's workstation poses security risks and restricts the user's electronic commerce activities to a single workstation. NetBill uses a key repository to optionally store customers' private keys. These keys are encrypted by a symmetric key derived from a passphrase known only to the customer.

4.1.1. Key Validation and Revocation Certificates

We use a public key certificate scheme (like that presented in [4]) to bind User IDs to keys, with NetBill as the certifying authority. NetBill generates a certificate when a customer first proves her identity and begins using NetBill.

However, allowing merchants, as services, to grant their own tickets based on these certificates poses a problem: NetBill is no longer involved in ticket-granting, and cannot prevent a ticket from being issued to a user with a compromised key. NetBill needs to invalidate compromised keys as quickly as possible.

NetBill maintains a Certificate Revocation List (CRL) at its server. When a key is compromised, the owner creates a Revocation Certificate and places it in the key repository along with her key. Any party can check that a given key has not been compromised by examining the revocation list.

Initially, it would seem that it is necessary for the customer and merchant to contact the server to check CRLs on each transaction. However, it is possible to

eliminate this check by allowing the NetBill transaction server to do it when it processes the payment transaction. By delaying the CRL check to late in the protocol, we introduce some minor risks. Customers and merchants may disclose information such as their preference for particular items or special prices to bogus peers, but there is no financial risk.

4.2. Pseudonyms

Some customers want to disguise their identities. NetBill provides two pseudonym methods to protect the privacy of the customer's identity: a per-transaction method that uses a unique pseudonym for each transaction, and a per-merchant method that uses a unique pseudonym for each customer-merchant pair. (See [3] for a full discussion of privacy protection with pseudonyms.) The per-merchant pseudonym is useful for customers who wish to maintain a consistent pseudonymous identity to qualify for frequent-buyer discounts.

These pseudonym schemes are implemented by introducing a pseudonym-granting server, *P*, to create pseudonymous PKTGTs for the customer. The protocol for obtaining and using a pseudonymous PKTGT is as follows. The customer obtains the pseudonymous PKTGT in steps 1–2, and uses it with a merchant in steps 3–4 exactly as she would use a normal PKTGT:

1. $C \Longrightarrow P$ $[\{\text{TrueIdentity, M, Timestamp, K1, Type}\}^{P}]_{C}$
2. $P \Longrightarrow C$ $E_{K1}(K2, [\{\text{Pseudonym, M, Timestamp, K2}\}^{M}]_{P}, [\text{TrueIdentity, M, Pseudonym, Timestamp}]_{P})$
3. $C \Longrightarrow M$ $[\{\text{Pseudonym, M, Timestamp, K2}\}^{M}]_{P}$
4. $M \Longrightarrow C$ $E_{K2}(T_{CM}(\text{Pseudonym}), CM)$

The protocol is the same for both kinds of pseudonyms; the desired type of pseudonym (per-merchant or per-transaction) is indicated in the *Type* field in step 1. The extra message *[TrueIdentity, M, Pseudonym, Timestamp]$_P$* in step 2 is the customer's receipt proving that she was using the pseudonym *Pseudonym* with the named merchant at the time indicated. This may be useful to the customer in conjunction with the receipt received in step 8 of the transaction (which contains only the pseudonym) to later prove that she was involved in the transaction.

5. Credentials and Authorizations

In [7], Neuman presents a system of using *restricted proxies* for authorization. A restricted proxy is a ticket giving the bearer authority to perform certain operations named in the ticket. NetBill uses a similar construct to implement *credentials* to prove group membership (to allow merchants to provide discounts to special groups) and to implement access control mechanisms.

5.1. Credentials for Group Membership

An organization can provide a credential server which issues credential proxies proving membership in a group. In this case, the credential server is asserting a fact (membership in a group) about which it is authoritative. For example, an auto club may provide a credential server which issues credentials to the members of the club; merchants who offer discounts to the club's members will accept these credentials as proof of membership. The protocol for obtaining a credential (assuming the customer has already obtained a service ticket for the credential server) from a credential server, *G*, is as follows:

1. $C \Longrightarrow G$ $T_{CG}(\text{Identity}), E_{CG}(\text{Group, CAcct})$
2. $G \Longrightarrow C$ $E_{CG}([\text{Group, Detail, Identity, CC(CAcct, AcctVN), Timestamp}]_{G}, \text{AcctVN})$

Credentials obtained in this manner are presented to merchants in the price request phase of the transaction protocol, step 1.

A credential issued to a customer may be unrestricted, or it may optionally be restricted for use on a specific account (for example, in order to prevent corporate employees from taking advantage of corporate discounts for personal purchases). This is accomplished by passing the account number to the group server as part of the request. If the account number is appropriate for this group, the credential will be issued. The credential contains a cryptographic checksum of the account number and an "Account Verification Nonce," which is also returned to the customer along with the credential.

This nonce is a pseudorandom number ensuring that merchants can neither determine which different customers (or the same customer in repeated sessions) are using the same account nor easily verify guesses of the customer's account number. The nonce is passed along to the NetBill server in the encrypted part of the EPO so that the NetBill server can verify that the checksum passed to the merchant (for his comparison to

the credential) corresponds to the account number actually being used.

The *Detail* field allows a credential server to include additional information in a format specific to the credential server. This would allow, for example, a multiple-journal subscription credential server to issue a single credential for all subscribers, using the *Detail* field to specify which journal subscriptions the customer holds.

Credentials can also be used by cooperating merchants to restrict information access. In this way, merchants only sell to approved customers: those who can present a certain credential. This offers a solution for merchants who, for example, can restrict distribution of sensitive documents only to individuals whose credentials verify a need-to-know.

5.2. Access Control Mechanism

As noted in [7], proxies can implement access control. An account owner (such as a parent) may have a restriction on the account such that no purchases can be completed by a given customer (such as a child) without approval from an access control server. This allows different organizations to provide access control services. For example, both the PTA and a church group could offer competing access control services.

To obtain an access control authorization, a customer *C* must present details of a specific transaction to the access control server *A*, who grants a single-use proxy authorizing the given transaction. The protocol is as follows:

1. $C \Longrightarrow A$ T_{CA}(Identity), E_{CA}(M, ProductID, Price, CC(E_K(Goods)), EPOID, CAcct)
2. $A \Longrightarrow C$ E_{CA}($E_{A\text{-}PRI}$(CC(Identity, M, ProductID, Price, CC(E_K(Goods), EPOID, CAcct)))

The item returned in step 2 is the *Authorization* item used in step 5 of the transaction protocol (see Section 3.4).

6. Complaints and Failure Analysis

The NetBill protocols are robust against failures, and retain essential information to protect customers and merchants against fraud. Our system can respond to complaints made by either the customer or the merchant. In this section, we examine those complaints and discuss how they are handled. First, we look at potential customer complaints, and then at potential merchant complaints.

6.1. Customer Complaints

6.1.1. Incorrect or Damaged Goods

- "This isn't the product I specified."
- "The goods arrived broken or incomplete."
- "The decryption key I was given was wrong."

In the event that the decrypted goods do not match the product description as given by the merchant, the dispute must be brought to the attention of a human arbitrator, who will determine the validity of the customer's complaint and, if appropriate, direct the merchant to provide a refund.

The arbitrator uses the registered copies at NetBill of the customer's signed EPO containing a cryptographic checksum of the encrypted goods, and the merchant's signed endorsement indicating his agreement with that cryptographic checksum and attesting to the decryption key. The arbitrator compares these registered values against the copy of the encrypted goods and decryption key provided by the customer in her complaint. The arbitrator can easily determine whether the purported problem with the goods is the fault of the merchant or an error by the customer.

- "The goods are not as advertised."

The protocol can be used to demonstrate whether the goods delivered are the goods ordered, as shown above. However, if the customer was induced to buy the goods by false advertising claims, this protocol provides no help. The customer must lodge a complaint with the Federal Trade Commission or other appropriate agency. It is important for billing servers to monitor these charges and assist with their resolution.

- "I bought this but never got the decryption key."

This complaint may be answered by directing the customer to perform a status query (see Section 3.5) to retrieve the key. In the event that the decryption key does not yield a satisfactory decryption, the dispute will change to one of the other complaints listed.

6.1.2. Transaction Disputes

- "I agreed to pay $X, but was charged $Y instead."
- "I've only bought $X worth of goods, but my balance has gone down by $Y."

Because the NetBill server has a signed EPO from the customer, it can prove that the customer approved

the purchase(s) for $Y. In the event that the NetBill server cannot provide the signed EPO(s), the customer's money is refunded. This protects customers against fraud by the operators of the NetBill server.

- "I never bought this, but it appears on my statement."
- "I told the merchant no, but he put it through anyway."

Because the NetBill server has signed EPOs from the customer, it can prove that the customer approved the purchases. In the event that the NetBill server cannot provide the signed EPOs, the customer's money is refunded.

- "You told me this transaction didn't go through, but I got charged anyway."

Because the NetBill server provides signed receipts even for failed transactions, the customer can present these receipts to prove that the transactions were declined. If the customer cannot produce these receipts and the NetBill server claims to have approved the transactions, it must provide the decryption keys for the information goods (via status query exchange).

6.2. Merchant Complaints

6.2.1. Insufficient Payment

- "I sold $X worth of goods but only received $Y."
- "You told me this transaction went through, but I never got paid for it."

In all transactions, the NetBill server provides a signed receipt indicating the success or failure of a transaction. In the event that a merchant is not properly credited, he can prove the error by presenting these signed receipts.

7. Conclusion

This paper has presented the NetBill protocols. These protocols have introduced new methods for certified delivery, access control, user certificates, pseudonyms, and their integration. These protocols are designed to provide very high degrees of security and flexibility while still providing good efficiency. However, this paper does not represent final work; it is a snapshot of our current design. We plan to test our design in a major wide-scale pre-commercial beta test of the NetBill system beginning in 1996.

The NetBill project is committed to open protocols. We are eager to work with others to make our protocols as widely applicable and interoperable as possible, and welcome comments.

For more information on the current state of the NetBill project, we invite readers to consult our WWW page at `http://www.ini.cmu.edu/netbill/`.

Acknowledgements

We received valuable contributions in technical discussions of the protocol from Thomas Wagner.

References

[1] Alireza Bahreman and J.D. Tygar. "Certified Electronic Mail." In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 3–19, San Diego, CA, February 1994.

[2] M. Bellare, *et al. iKP Family of Secure Electronic Payment Protocols.* `http://www.zurich.ibm.com/Technology/security/extern/ecommerce`

[3] Benjamin Cox. *Maintaining Privacy in Electronic Transactions.* Information Networking Institute Technical Report TR 1994–8, Fall 1994.

[4] Stephen Kent. *RFC 1422: Privacy Enhancement for Electronic Mail: Part II: Certificate-Based Key Management.* Internet Activities Board Request For Comments 1422, February 1993.

[5] National Institute of Standards and Technology. *FIPS 180: Federal Information Processing Standard: Secure Hash Standard (SHS).* April 1993.

[6] National Institute of Standards and Technology. *FIPS 186: Federal Information Processing Standard: Digital Signature Standard (DSS).* May 1994.

[7] B. Clifford Neuman. "Proxy-Based Authorization and Accounting for Distributed Systems." In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, May 1993.

[8] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In *Communications of the ACM*, 21(2), February 1978.

[9] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* New York: John Wiley & Sons, 1994.

[10] Marvin Sirbu and J.D. Tygar. "NetBill: An Internet Commerce System Optimized for Network Delivered Services." In *IEEE Personal Communications*, pages 6–11, August 1995.

[11] Alexander Somogyi, Thomas Wagner, *et al. NetBill.* Information Networking Institute Technical Report TR 1994–11, Fall 1994.

[12] Jennifer G. Steiner, B. Clifford Neuman and Jeffrey I. Schiller. "Kerberos: An Authentication Service for Open Network Systems." In *USENIX Winter Conference*, pages 191–202, February 1988.

[13] J. D. Tygar. "Atomicity in Electronic Commerce" (invited paper), to appear in *ACM/IEEE 21st Conference on Principles of Distributed Computation*, 1996.

*i*KP – A Family of Secure Electronic Payment Protocols

EXTENDED ABSTRACT*

Mihir Bellare†, Juan A. Garay†, Ralf Hauser‡, Amir Herzberg†,
Hugo Krawczyk†, Michael Steiner‡, Gene Tsudik‡, Michael Waidner‡

August 2, 1995

Abstract

This paper proposes a family of protocols – *i*KP ($i = 1, 2, 3$) – for secure electronic payments over the Internet. The protocols implement credit card-based transactions between the customer and the merchant while using the existing financial network for clearing and authorization. The protocols can be extended to apply to other payment models, such as debit cards and electronic checks. They are based on public-key cryptography and can be implemented in either software or hardware. Individual protocols differ in key management complexity and degree of security. It is intended that their deployment be gradual and incremental.

The *i*KP protocols are presented herein with the intention to serve as a starting point for eventual standards on secure electronic payment.

1 Introduction

Nowadays it is hardly necessary to stress the importance of electronic commerce. Let us just note that it is rapidly gaining momentum, and is equally appealing to both (electronic) merchants and consumers.

There is widespread agreement that to enable electronic commerce one needs means for *secure electronic payments*. Indeed, the appeal of electronic commerce without electronic payment is limited. Moreover, *insecure* electronic payment methods are more likely to impede, than to promote, electronic commerce.

In this paper we propose a family of secure electronic payment protocols – *i*KP (*i*-Key-Protocol, $i = 1, 2, 3, \ldots$). The protocols are compatible with the existing business models and payment system infrastructure. They involve three parties: the customer (who will make the payment), the merchant (who will receive the payment) and a party called a *gateway* (who acts as a gateway between and electronic world and the existing payment infrastructure, and will authorize the transaction by using the existing infrastructure).

Within this framework we focus on the credit card payment model as it is anticipated to be the most popular in the near future. (The customer has a credit card. The gateway is now called an acquirer gateway and will use the existing credit card transaction verification infrastructure to authorize requests). However, the protocols can be extended to apply to other payment models, e.g., debit cards and electronic checks.

All *i*KP protocols are based on public-key cryptography, but they vary in the number of parties (out of the three involved parties) that possess their own public key pairs. This number is indicated by the name of the individual protocols: 1KP, 2KP, and 3KP. The *i*KP protocols offer increasing levels of security and sophistication as the number of parties who can hold public key pairs increases.

The simplest protocol, 1KP, requires that only the acquirer gateway possess a pair of public and private keys. Customers and merchants need only to

† IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA, {mihir, garay, amir, hugo}@watson.ibm.com

‡ IBM Zürich Research Laboratory, Sämerstrasse 4, CH-8803 Rüschlikon, Switzerland, {rah, sti, gts, wmi}@zurich.ibm.com

* The most recent version of this document is available from <http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/>.

possess the authentic public key of the gateway, or the authentic public key of an "authority" that validates the gateway's public key via a signed certificate. This involves a minimal certification authority infrastructure that provides certificates for a small number of entities, namely, the acquirer gateways. Such a minimal certification authority can be run, for example, by the credit card company itself. (In which case, a customer will keep, say, the "VISA public key" or "MasterCard public key", etc.) In the 1KP scenario, customers are authenticated on the basis of their credit card numbers, and possibly associated secret PINs. Payments are authenticated by communicating the credit card number and PIN appropriately *encrypted* under the acquirer's public key, and properly bound to relevant information (purchase amount, id's, etc.). While 1KP is very simple, it does not offer non-repudiation for messages sent by customers and merchants; this means that disputes about the authenticity of payment orders are not easily resolvable. Some consequences of missing non-repudiation for payment systems are illustrated in [1].

2KP asks that merchants, in addition to acquirer gateways, hold public key-pairs and public key certificates. The protocol is can then provide non-repudiation for messages originated by merchants. Additionally, 2KP enables customers to verify that they are dealing with *bona fide* merchants by checking their certificates, without any on-line contact with a third party. As in 1KP, payment orders are authenticated via the customer's credit card number and PIN (encrypted before transmission).

3KP further assumes that the customer has a public key pair, and then provides full multi-party security. It achieves non-repudiation for all messages of all parties involved. Payment orders are authenticated both by the credit card number and PIN, and a digital signature of the customer. This makes the forging of payment orders computationally infeasible. Additionally, 3KP enables merchants to authenticate customers on-line. Notice that in this case a full certification authority infrastructure is required to provide certificates of the customer's public keys.

All *i*KP protocols can be implemented either in software or hardware. In fact, in 1KP and 2KP the customer does not even need a personalized payment device: only credit card data (and the PIN if present) must be entered to complete a payment. However, for the sake of increased security, it is desirable to use a tamper-resistant device that can protect the PIN and – in case of 3KP – the secret key of the customer.

As public key technology becomes more pervasive, we expect more and more parties to hold public key pairs. A gradual deployment of the *i*KP protocols thus makes sense: begin with 1KP, then move to 2KP and finally to 3KP.

We stress that the job of *i*KP is to enable *payments*. It is not concerned with any aspect of the determination of the order; it assumes that the order, including price, have already been decided on between customer and merchant.

The iKP protocols do not explicitly provide encryption of the order information. Such protection is assumed to be provided by other existing mechanisms, e.g., SHTTP [18] or SSL [14]. The decoupling of order encryption from the electronic payment protocol is an important design principle of *i*KP which supports compatibility with different underlying browsing and privacy-protecting mechanisms. It also adds to the simplicity, modularity, and ease of analysis of the protocols. An additional advantage is freeing *i*KP from export restrictions related to the use of bulk encryption. Nonetheless, if desired, the *i*KP family (especially, 2KP and 3KP) can be easily extended to generate shared keys between customer and merchant for protection of browsing and order information.

2 Related Work

The *i*KP family has many features and motivations in common with other proposals for on-line payment systems[1].

Most proposals for on-line payment are based on standard models (e.g, credit cards) and connect the electronic and the conventional payment system via some sort of gateway (e.g., [20, 9, 12]).

As already mentioned, most current on-line payments are not protected at all. Some systems propose symmetric cryptography for efficiency reasons (e.g., [11, 19, 17]), in particular, those that aim at micro-payments. However, most proposals use public key cryptography in a way similar to one or another *i*KP protocol. For example, [12] uses public-key cryptography between merchant and gateway, and the protocol sketched in [9] appears cryptographically similar to 2KP.

The most cryptographically advanced electronic payment systems emphasize *untraceability* and *anonymity* against the payment system [3, 5, 6, 4, 10, 16].

Finally, there are some general security schemes

[1] For a comprehensive listing see [15] or <http://www.zurich.ibm.com/ Technology/Security/sirene/outsideworld/ecommerce.html>.

for the *World Wide Web*, most notably, SHTTP [18] and SSL [14]. Both have been suggested as a basis for secure electronic payments. SHTTP is a possible platform for implementing *i*KP. SSL is more thought to secure the link between WWW client and server, and is therefore less suited for multi-party protocols like *i*KP. Moreover, SSL does not support non-repudiation.

The particular advantages of *i*KP over existing proposals are:

- The *i*KP family allows for a gradual deployment. 1KP is based on what already exists today – credit cards, PINs, and the existing payment system networks – and presents a feasible short-term solution. Introduction of public key cerfitication of merchants will usher in 2KP and, as soon as the certification infrastructure for customers is in place, 3KP can be phased in and achieve full multi-party payment security.
- *i*KP is an evolving design, rather than a fixed, closed protocol. It is intended as a starting point for a standard for secure payments over the Internet. It is open for discussion, and we encourage comments on its qualities and suggestions for improvement.
- The use of encryption in *i*KP is limited to well-defined payment data – credit card numbers and PINs – and the interfaces to cryptographic primitives can be designed in a way that makes them inaccessible to the end-user. Therefore, we expect that *i*KP (at least specific implementations as sketched in [13]) will not be subject to US export regulations. We note that, for countries outside North America, there is absolutely no incentive to use payment systems with restricted or weakened security.

A specific software-only architecture used for implementing a prototype of *i*KP is described in [13]. It is independent of any HTTP-extension and works with any WWW browser.

3 Payment Model

PARTIES. All *i*KP protocols are based on the existing credit-card payment system. The parties in the payment system are shown in Figure 1.

The payment system is operated by a *payment system provider* like Europay, MasterCard, VISA. This payment system provider has fixed business relations with certain banks who act as *issuers* of credit cards to customers, or as *acquirers* of payment records from merchants. Each issuer has a Bank Identification Number, BIN, which it receives at the time it signs up with a payment system provider, and which is embossed on each credit card issued as part of the credit card number. The BIN also identifies the payment system provider.

A *customer* receives a credit card from an issuer, and is in possession of a PIN as is common in current systems. In 1KP and 2KP, payments will be authenticated only by means of the credit card number and this PIN (both suitably encrypted!), while in 3KP, a digital signature is additionally used.

It is assumed that (as can be expected for electronic payment) that the customer is using a computer to execute the payment protocol. Since this computer must receive the customer's PIN or secret signature key, it must be a trustworthy device. We caution that even a customer-owned computer is vulnerable: it may be used by several persons or it may contain a Trojan horse or a virus that could steal PINs and secret keys. The best payment device would be a secure isolated computer, e.g., a tamper-resistant smartcard, connected to the computer used for shopping via a customer-owned smartcard reader with its own keyboard and display. (This is often called an *Electronic Wallet.*) Technically, 1KP and 2KP can be used with any kind of payment device, while for 3KP the customers need personal devices that store their secret signature keys and certificates.

A *merchant* signs up with the payment system provider and with a specific bank, called an *acquirer,* to accept deposits. Like a customer, a merchant needs a secure device that stores the merchant's secret keys and performs the payment protocol.

Clearing between acquirers and issuers is done using the existing financial networks.

The *i*KP protocols deal with the *payment* transaction only (i.e., the solid lines in Figure 1), and therefore involve only three parties, called C – Customer, M – Merchant, and A – Acquirer Gateway. Note that A is no acquirer in the financial sense, but a *gateway* to the existing credit card clearing/authorization network. In other words, the function of A is to serve as a front-end to the *current infrastructure that remains unchanged.*

The protocols presented here describe the core of a payment system only. Besides this, additional mechanisms are needed, e.g., for *cancelation of payment orders* and for *providing statements of account.*

PUBLIC KEYS AND CERTIFICATION. Since all *i*KP protocols are based on public-key cryptography, we need a mechanism to authenticate these public keys. We assume a *certification authority, CA*, which has a secret key, SK_{CA}. Its public counterpart, PK_{CA}, is

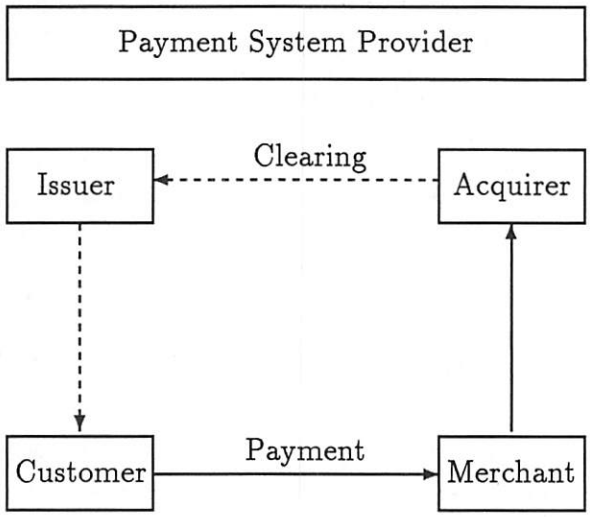


Figure 1: Generic Model of a Payment System

held by all other parties. CA will certify a public key of party X by signing the pair (X, PK_X) consisting of the identity of X and X's public key. (The signature is computed under SK_{CA}.) Note that PK_{CA} must be conveyed in an authenticated manner to every party. This will be typically done out-of-band, via any of a number of well-known mechanisms.

For simplicity's sake, we assume that there is only one certification authority. However it is easy to extend the protocols to support multiple certification authorities, e.g., such that the payment system provider at the top-level authority issues certificates to its constituent issuers and acquirers, while these, in turn, issue certificates to their customers and merchants.

In all iKP protocols, an acquirer A has a secret key, SK_A, which enables signing and decryption. Its public counterpart, PK_A, (which enables signature verification and encryption) is held by each accredited merchant together with its corresponding CA's certificate. As in current operation, acquirers receive the customers' credit card numbers and PINs, and are trusted to keep these values confidential.

In 2KP, each merchant, and in 3KP also each customer, have a secret/public key-pair. They are denoted by $(\mathrm{SK}_M, \mathrm{PK}_M)$ and $(\mathrm{SK}_C, \mathrm{PK}_C)$, respectively. Both public keys are included in certificates issued by CA.

ADVERSARIES AND THREATS. We consider three different adversaries:

- **Eavesdropper** who listens to messages and tries to learn secrets (e.g., credit card numbers, PIN's)
- **Active attacker** who introduces forged messages in an attempt to cause the system to misbehave (e.g., to send him goods instead of to the customer)
- **Insider** who either is some legitimate party or learns that party's secrets. (One example is a dishonest merchant who tries to get paid without authorization.)

Before listing requirements in Section 4 we briefly discuss common threats and attacks.

The *Internet* is a decentralized, heterogeneous network, without single ownership of the network resources and functions. In particular, one cannot exclude the possibility that messages between the legitimate parties would pass through a maliciously controlled computer. Furthermore, the routing mechanisms in the Internet are not designed to protect against malicious attacks. Therefore, it is folly to assume either confidentiality or authentication for messages sent over the Internet, unless proper cryptographic mechanisms are employed. To summarize, it is easy to steal information off the Internet. Therefore, at least credit card numbers and PINs must *not* be sent in the clear.

In addition, one must be concerned about the trustworthiness of the merchants providing Internet service. The kind of business that is expected in the Internet would include the so-called *cottage industry* – small merchants. It is very easy for an adversary to set up shop and put up a fake electronic *storefront* in order to get customers' credit card numbers. This

implies that the credit card number should travel from customer to acquirer without being revealed to the merchant (who needs only the BIN which can be provided separately.)

Obviously, a good deal of care must be taken to protect the keys of acquirers. One of the biggest concerns is that of an adversary breaking into an acquirer computer through the Internet connection. Therefore, the acquirer's computer must be protected with the utmost care; including a very limited Internet connection using advanced firewall technology (e.g., [8, 7].)

Furthermore, the trust in the acquirer's computer must be limited, so that a break in would have a limited effect only.

4 Security Requirements

In this section we consider a range of potential requirements for each party involved in the payment process: issuer/acquirer, merchant, and customer. They range from mandatory security requirements to optional features.

ISSUER/ACQUIRER REQUIREMENTS. The issuer and the acquirer are assumed to enjoy some degree of mutual trust. Moreover, an infrastructure enabling secure communication between these parties is already in place. Therefore, we join the requirements of the issuer and the acquirer.

A1– *Proof of Transaction Authorization by Customer.* When the acquirer debits a certain credit card account by a certain amount, the acquirer must be in possession of an unforgeable *proof* that the owner of the credit card has authorized this payment. This proof must not be "replayable," or usable as proof for some other transaction. This means it must certify at least the amount, currency, goods description, merchant identification, and delivery address, and be obtained in such a way that replay is not possible. (We use a combination of time stamps and nonces for this purpose). Note also that in this context the merchant may be an adversary, and even such a merchant must not be able to generate a fake debit. We distinguish between:

(a) *Weak Proof*, which authenticates the customer to the acquirer but does not serve as a proof for third parties, and

(b) *Undeniable Proof*, which provides full non-repudiation, i.e., can be used to resolve disputes between the customer and the payment system provider.

The same distinction will be made for all subsequently required proofs of transaction.

A2– *Proof of Transaction Authorization by Merchant.* When the acquirer authorizes a payment to a certain merchant, the acquirer must be in possession of an unforgeable *proof* that this merchant has asked that this payment be made to him.

MERCHANT REQUIREMENTS. We ask for two guarantess for the merchant.

M1– *Proof of Transaction Authorization by Acquirer.* The merchant needs an unforgeable proof that the acquirer has authorized the payment. This includes certification and authentication of the acquirer, so that the merchant knows he is dealing with the real acquirer, and certification of the actual authorization information. Note that again the amount and currency, the time and date, and information to identify the transaction must be certified. We also distinguish between (a) *Weak proof* and (b) *undeniable proof*, which provides full non-repudiation.

M2– *Proof of Transaction Authorization by Customer.* Even before the merchant receives the transaction authorization from the acquirer, the merchant might need an unforgeable proof that the customer has authenticated it. Again we distinguish between (a) *Weak Proof* and (b) *Undeniable Proof.* This requirement is necessary to provide for *off-line authorization.*

CUSTOMER REQUIREMENTS. We ask for the following guarantess to the customer who is making the payment.

C1– *Unauthorized Payment is Impossible.* It must not be possible to charge something to a customer's credit card without possession of the credit card number, PIN, and in case of 3KP, the customer's secret signature key. Thus, neither Internet rogues nor malicious merchants must be able to generate spurious transactions which end up approved by the acquirer. This must remain the case even if the customer has engaged in many prior legitimate transactions. In other words, information sent in one (legitimate) transaction

must not enable a later spurious transaction. So in particular the PIN must not be sent in the clear, and not even be subject to guessing attacks! Similar to the two type of proofs of transactions, we distinguish between:

(a) *Impossibility*, which means that unauthorized payments are impossible provided the acquirer's secret key is not available to the adversary, and

(b) *Disputability*, which means that even if the acquirer's secret key is available to the adversary (e.g., because the adversary co-operates with an insider), the customer can prove that he/she did not authorize the payment.

In fact, these two requirements are typically met by meeting the corresponding acquirer requirements A1.a and A1.b, respectively.

C2– *Proof of Transaction Authorization by Acquirer.* The customer would like to be in possession of proof that the acquirer authorized the transaction. This "receipt" from the acquirer is not of paramount importance, but is convenient to have. Again, we distinguish between (a) *Weak Proof* and (b) *Undeniable Proof* (full non-repudiation).

C3– *Certification and Authentication of Merchant.* The customer needs a proof that the merchant is accredited at an acquirer (which could be considered as some guarantee for the trustworthiness of the merchant).

C4– *Receipt from Merchant.* The customer wants a proof that the merchant who has made the offer has received payment and promised to deliver the goods. This takes the form of an undeniable receipt. 2KP and 3KP will satisfy this requirement, but will not ensure fairness: The merchant can always refuse sending this receipt while already having received the authorization message from the acquirer gateway. In this case, the customer must take the next statement of account as a replacement for this receipt.

ADDITIONAL POSSIBLE REQUIREMENTS. Requirements C1 – C4 are discussed in the following. Other requirement that may be desirable, but are not explcitly addressed here; however we now discuss these requirements and their relation to *i*KP.

C5– *Privacy.* Customers want privacy of their order and payment information. For example a businessman may be purchasing the latest information on certain stocks and may not want competitors to know which stocks he is interested in. The privacy of order information and amount of payment should be implemented independently of the payment protocol, e.g., based on SHTTP [18] or SSL [14]. *i*KP does provide some privacy: it does not reveal order information to any other party than the merchant, at least as long as there is no dispute. But does not include encryption of these data. Obviously, credit card number and PIN must be protected carefully, which is achieved within *i*KP by encrypting them with the acquirer's public key. (This is the only application of *encryption* in *i*KP, which is made in order to facilitate exportability from the US.)

C6– *Anonymity.* Besides confidentiality of order and payment information, customers may want *anonymity* from eavesdroppers and (optionally) also from the merchant. It is also conceivable that the customer may even want anonymity with respect to the payment system provider. *i*KP supports anonymity from the merchants in the sense that the customer's identity, address, etc., is not revealed to the merchant: the customer uses a pseudo-identity CID which is different in each transaction. *i*KP does not offer anonymity from the payment system provider. This might be desirable for systems that aim to imitate cash, but is not essential for protocols, like *i*KP, that follow the credit card-based payment model.

5 The *i*KP Family

PRIMITIVES AND KEYS. Figure 2 summarizes the notation for the keys held by the various parties, and the cryptographic primitives we will be using. While A's key pair must enable signature and encryption, all other key pairs need to enable signatures only. Note that signing and encryption are *independent* operations; in particular, $\mathcal{E}_X(\mathcal{S}_X(\alpha)) \neq \alpha$.

We want an encryption function $\mathcal{E}_X$ which provides some form of "message integrity." Decryption of a ciphertext results either in a plaintext message, or in a flag indicating non-validity. Formally, the primitive we has the property that correct decryption convinces the decryptor that the transmitter "knows" the plaintext that was encrypted. In particular, tampering with ciphertext is detectable. A

- **Keys:**

$\mathrm{PK}_X, \mathrm{SK}_X$	Public and secret key of Party X (X = Certification Authority CA, Customer C, Merchant M, Acquirer Gateway A).
CERT_X	Public key certificate of Party X, issued by CA. We assume it includes X, PK_X and CA's signature on PK_X.

All protocols assume A has a public key, and any party needing it has PK_{CA}. 1KP assumes no other keys; 2KP additionally assumes M has a public key; 3KP further assumes C also has a public key.

- **Cryptographic primitives:**

$\mathcal{H}(\cdot)$	A strong collision-resistant one-way hash function. Think of $\mathcal{H}(\cdot)$ as returning "random" values.
$\mathcal{E}_X(\cdot)$	Public-key encryption using PK_X, done in a way to provide not only confidentiality but also some kind of "message integrity."
$\mathcal{S}_X(\cdot)$	Signature with respect to SK_X. Note the signature of message M does NOT include M. We assume the signature function hashes the message before signing.

Figure 2: *Keys and cryptograhic primitives used in iKP protocols*

simple scheme to achieve such *plaintext-aware encryption* using RSA is described in Appendix A, based on the scheme of [2].

We stress that such encryption does not provide authentication in the manner of a signature, i.e., it does not provide non-repudiation. But it can be made to provide an authentication like capability between parties sharing a key (such as the CAN or PIN).

We note that the encryption function is *randomized:* $\mathcal{E}_X$ invoked upon message m will use, to compute its output, some randomizer, so that each encryption is different from previous ones.

A prototype implementation done at IBM Research uses RSA with key length 1024 for signature, and for the basis of the plaintext aware encryption; it also uses MD5 as a hash function.

Figure 3 is a list of quantities that will occur in the protocols. Their meaning and usage will be further explained as we go along.

FRAMEWORK OF *i*KP PROTOCOLS. The protocols have a common framework. Figure 4 illustrates the flows at a very high level. Before the protocol begins, each party $X \in \{A, C, M\}$ has some starting information represented by $\mathrm{ST\text{-}INF}_X$. The customer starts with the public key PK_{CA} of the certification authority. The merchant has the certificate CERT_A of the acquirer, and the acquirer has his own certificate CERT_A plus the corresponding secret key SK_A. They may each also have other information, which differs depending on whether we are in 1KP, 2KP or 3KP, and will be specified at the appropriate time.

It is assumed that before the protocol starts, the customer and merchant have agreed on the description and price of the items to buy. The functionality required to shop and agree on the item and price are to be provided by the browser, not by *i*KP. Thus DESC and PRICE are part of the starting information of merchant and customer.

The protocol consists of five flows. The exact content of these flows depends on the protocol: they are different in 1KP, 2KP and 3KP. At a high level, however, there is a common structure. The customer starts with an Initiate flow. The merchant responds by providing the Invoice. The customer than makes the Payment which the merchant uses to send an authorization request Auth-Request to the acquirer. The acquirer goes through the financial network to obtain the authorization and returns an authorization response Auth-Response to the merchant. The latter processes this to produce a confirmation flow Confirm for the client.

The main difference between 1KP, 2KP and 3KP is the increasing use of signatures as more of the parties involved are able to use public keys.

5.1 1KP

1KP (illustrated in Figure 5) represents the initial step in the gradual introduction of a public-key infrastructure. Although it requires the use of public-key encryption by all parties, only the acquirer gateway, A, needs to possess and distribute its own public key certificate, CERT_A. In particular, the total number of certificates to be issued by the certification authority is small as it depends only on the number of gateways.

Like all members of the *i*KP Family, 1KP requires that all customers and merchants have an authentic copy of PK_{CA}, the public key of the certification authority. A customer C has a customer account number CAN known to the acquirer. This could be a credit card number. It may also have a secret PIN which is also known to the payment system (but not to the merchants!). Every merchant has to know the certificate of the corresponding acquirer gateway, CERT_A.

1KP does not assume A to keep a state per customer. Instead, the customer's PIN is verified using the existing authorization infrastructure (which uses tamper-resistant technology for processing and verification of PIN's).

All parties in 1KP must perform certain public key computations. Encryption is only applied *once,* for sending credit card data and PIN from the customer to the acquirer gateway, securely. Therefore, public key encryption is required from C only, while decryption is required from A only (this is true also for 2KP and 3KP). In 1KP, only A has to sign some data, which must be verified by C and M. We now provide the flow by flow actions of the parties.

Initiate: Customer forms CID by generating random number R_C and computing $\text{CID} = \mathcal{H}(R_C, \text{CAN})$. Generates another random number SALT_C to be used for "salting" the hash of merchandise description (DESC) in subsequent flows. Sets Text_0 to include desired protocol options (if any) and/or DESC. Sends Initiate.

Invoice: The computation of the second flow, Invoice, takes place as follows. Merchant retrieves SALT_C and CID from Initiate. Chooses/obtains DATE– this is a time stamp, and indicates, say the hour as well. Generates nonce NONCE_M. The combination of DATE and NONCE_M will be used later by A to uniquely identify this order: the nonce disambiguates payments with a common DATE. Chooses transaction id TID_M which identifies the context. Computes $\mathcal{H}(\text{DESC}, \text{SALT}_C)$. Forms Common as defined above and computes $\mathcal{H}(\text{Common})$. Note: seller does not need to additionally "salt" $\mathcal{H}(\text{Common})$ because it contains the already-salted $\mathcal{H}(\text{DESC}, \text{SALT}_C)$. Composes Text_1. (If C did not already have CERT_A then it could go here. Or this could include a context pointer for the customer.) Finally sends Invoice.

Payment: Customer retrieves Clear from Invoice. He retrieves ID_M, DATE, TID_M and NONCE_M. He already has PRICE and CID, so that he can now form Common. He computes $\mathcal{H}(\text{Common})$ and checks that this matches the value in Clear. He then forms the SLIP as defined in Figure 5. (It includes the price, the customer account number (credit card number), and $\mathcal{H}(\text{Common})$. It also includes the salt R_C used to form the CID, and optionally the PIN if present.) The slip is now encrypted under the acquirer public key: he sets $\text{EncSlip} = \mathcal{E}_A(\text{SLIP})$. This, along with the optional Text_2, is the Payment flow sent to the merchant. (Note: Customer doesn't do any check on DATE, other, perhaps, than making sure it is not in the future! The DATE goes into Common, and SLIP contains $\mathcal{H}(\text{Common})$, and the acquirer will check the latter.)

Auth-Request: The merchant will now ask check that the acquirer authorizes the payment. He forwards EncSlip. He also sends Clear and $\mathcal{H}(\text{DESC}, \text{SALT}_C)$, and optional Text_3.

Auth-Response: The acquirer gateway extracts Clear, $\mathcal{H}(\text{DESC}, \text{SALT}_C)$ and EncSlip from Auth-Request. It then does the following:

(1) Extracts from Clear the following– ID_M, TID_M, DATE, NONCE_M and the value h_1 which is supposed to be $\mathcal{H}(\text{Common})$. It now checks for replays. That is, it makes sure that there is no previously processed request with these values of ID_M, TID_M, DATE and NONCE_M.

(2) Now it decrypts EncSlip. If the decryption fails, then the alteration of EncSlip (by an adversary or by M) is detected and the transaction is invalid. If not, A gets SLIP. Now A extracts PRICE, the value h_2 which is supposed to be $\mathcal{H}(\text{Common})$, CAN, R_C, and, if present, the PIN from SLIP.

(3) It checks that $h_1 = h_2$– this ensures that merchant and customer agree on the order information (price, identity of merchant, etc).

(4) It re-forms Common. (It has PRICE from SLIP. It has ID_M, TID_M, DATE, and NONCE_M from Clear. It can compute $\text{CID} = \mathcal{H}(\text{CAN}, R_C)$ because it has R_C and CAN

from SLIP. Finally it has $\mathcal{H}(\text{DESC}, \text{SALT}_C)$ from Auth-Request. These put together yield Common.) It then computes $\mathcal{H}$(Common) and checks this equals the value $h_1 = h_2$ above.

(5) Now it uses the existing clearing and authorization system to on-line authorize the payment: for this, it will forward CAN, PIN if present, the price, etc as dictated by the authorization system. Upon receipt of a response Y/N from the authorization system, A computes a signature, using the function $\mathcal{S}_A$, on Y/N and $\mathcal{H}$(Common).

Finally it sends Auth-Response and possibly Text_4. The latter could include TID_M so that the merchant can easily recover the context.

Confirm: M receives Auth-Response. He extracts Y/N and the acquirer signature. He already has $\mathcal{H}$(Common). Now he checks that the acquirer sent a valid signature of Y/N, $\mathcal{H}$(Common). He then forwards Y/N to the customer. He also forwards the acquirer signature so that the customer may check it.

There are some final checks by the buyer: for example it may want to check the acquirer signature. We stress here that the use of $\mathcal{H}$(Common) in the signature (as opposed to using the explicit values amount, currency, etc.), is done in order to protect the privacy of these data when transmitted to merchant and customer. We now look at which requirements 1KP satisfies.

A1(a) *Proof of Transaction Authorization by Customer.* SLIP includes the CAN and the PIN. (The latter, if present, is known only to the customer and payment system and is the basis of the security. If it is not present, one must assume the CAN is not known to an adversary.) Since C knows PK_{CA} and verifies CERT_A, it is ensured that C does not unwittingly send the CAN and PIN to a non-authorized party. A decrypts and checks that the CAN and PIN are correct. The plaintext-awareness of the encryption (see beginning of Section 5) implies that SLIP originated with the CAN and PIN holder. An adversary not knowing the CAN or PIN can neither create a fake SLIP nor modify the encryption of a legitimate one to its advantage.

Replay of a SLIP by a dishonest merchant will be detected by the combination of the DATE and NONCE_M. There is an "acceptable delay" period T_{delay}. Slips containing a particular DATE are kept until for T_{delay} more time than that indiacated by DATE. (For example, DATE could be the date and hour, and the delay period a day, meaning slips are kept for a day more than the DATE marked on them.) Within a particular value of DATE, different slips are disambiguated by the nonces.

The "semantic security" of the encryption (and in particular the fact that it is randomized) implies also security against *dictionary-attacks.* If the attacker knows all data in SLIP except PIN, he could compute encryptions $\mathcal{E}_A$(SLIP) for *all* possible values of PIN. With a deterministic encryption function, he could easily determine the correct PIN by comparing all encryptions with the one produced by C. Therefore, plain-text aware encryption is randomized: if SLIP is encrypted twice, two *different* cyphertexts are produced, which excludes this type of attack.

Note that PIN-based authentication provides a weak proof only. Signature-based authentication as used in 3KP provides an undeniable proof. Moreover, the probability of guessing the correct PIN is much higher than the probability of guessing a valid-looking signature.

It is important to stress that the "transaction" that of which we want a proof includes the item description, and in particular the delivery address. It should not be possible for an adversary to divert a legitimate payment by changing the delivery address. The inclusion of $\mathcal{H}$(Common) in A's authorization is to prevent such attacks. In particular it prevents a certain kind of *man-in-the-middle* attack that we now describe.

An attacker that impersonates a merchant can get the agreement of the customer to buy something for a given amount. The adversary gets from the customer an encrypted slip authorizing the payment. The adversary now impersonates the customer to the merchant, but this time the adversary buys for the same amount a (possibly) different merchandise with different delivery address and "pays" for it with the customer's slip. Notice, however, that in this case there will be a mismatch between the view of the "order" by the real customer and the merchant, and, consequently, a mismatch in the value of $\mathcal{H}$(Common).

M1(b): *Proof of Transaction Authorization by Acquirer.* The unforgeable, undeniable proof is the digitally-signed message sent by A. Notice that we have used a digital signature so that non-repudiability is provided. The inclusion of $\mathcal{H}$(Common) prevents the replay of authorization messages which would result in fake authorization of customer's orders.

Since the merchant knows Common in advance, the signature would indicate any tampering in the information sent from merchant to acquirer, and any

disagreement between customer and merchant on the payment data.

The inclusion of $\mathcal{H}$(Common) both in the customer-generated SLIP and directly in Auth-Request by the merchant enables A to detect a disagreement between merchant and customer with respect to the order contents (even before submitting the transaction to the clearing network).

C1(a): *Unauthorized Payment is Impossible.* This is a direct consequence of the achievement of A1(a).

C2(b): *Proof of Transaction Authorization by Acquirer.* As for M1(b).

C5: *Privacy.* Some partial privacy is provided. Specifically, the acquirer is not given DESC, but rather $\mathcal{H}(\text{DESC}, \text{SALT}_C)$. Furthermore, the acquirer, or an eavesdropper on the acquirer to merchant link, cannot obtain DESC via a dictionary attack, as we now explain.

In a dictionary attack, the attacker has some small set of possible values of DESC, and want to see whether one of them is what the customer is ordering. Had we not used the salt, but just sent $\mathcal{H}$(DESC), the attacker could easily make the check by evaluating $\mathcal{H}$ on his values and seeing whether one of the results matches the value $\mathcal{H}$(DESC) in the flow. But now he cannot do this because he doesn't know SALT_C. Of course, if he was powerful enough to obtain SALT_C of the C to M link (where it was transmitted in the clear) he would be able to do the dictionary attack, but that he can eavesdrop like that on both links is not too likely. Also, if privacy is really a concern, the C to M communication may be protected by SSL or SHTTP.

We stress that provision of privacy is not a primary concern of a payment protocol. However, we wish at least to not give anything away that should not be, and took the chance to add whatever privacy we could add without much cost.

The last flow from merchant to customer in which the signed authorization by the acquirer is transmitted is optional. It only serves as a receipt for the customer but is not needed for the security of the payment protocol.

To summarize, 1KP is a simple and efficient protocol whose main achievement is to get a secure electronic payment system with as little modification as possible to the existing infrastructure. Its main weaknesses are: 1) the customer authenticates itself via the acquirer and only using a credit card number and PIN (as opposed to a strong authentication via a digital signature); 2) the merchant does not directly authenticate itself to the customer or acquirer (there is some level of indirect authentication via the customer's SLIP and the authorization by the acquirer); and 3) neither merchant nor customer provide undeniable receipts for the transaction. Upgrading 1KP to provide these missing features results in the protocols described in the next two subsections, namely, 2KP and 3KP.

5.2 2KP

The second protocol, 2KP, is illustrated in Figure 6. The basic difference with respect to 1KP is that, in addition to A, each merchant M needs to possess a public key with a matching secret key, and distribute its own public key, with its certificate, CERT_M.

We now describe the additions to the flows and actions. There are two new elements in Invoice. The first is that the merchant chooses a random V and puts $\mathcal{H}(V)$ in Invoice. (The inclusion of V in Confirm will later serves as a "signature" thereby saving the merchant one signature computation. See below.) This value will be added to Common for what follows. Second, the merchant signs (using SK_M) the pair of strings $\mathcal{H}$(Common) and $\mathcal{H}(V)$ and includes this signature Sig_M in Invoice too. Furthermore the merchant includes CERT_M so that the customer can check his signature. Upon receipt of Invoice the customer checks the merchant signature, and then proceeds as before to generate Payment. Auth-Request is augmented by the merchant to include the same signature Sig_M he sent to the customer earlier, together with CERT_M. The acquirer checks this signature before authorizing payment. Finally, the value V is included by the merchant in Confirm. The customer computes $\mathcal{H}(V)$ and checks that it matches the value sent earlier in Invoice.

2KP satisfies all the requirements addressed by 1KP as well as:

A2: *Certification and Authentication of Merchant.* This is achieved by the inclusion of the merchant signature Sig_M and certificate CERT_M, and the acquirer's verification of these.

C3: *Certification and Authentication of Merchant.* Similarly achieved by inclusion of signature of merchant and its check by customer.

C4: *Receipt from merchant.* This is achieved by the combination of M's signed message sent to A, A's signed authorization message, and the value V sent in confirm. V assures the customer (and any third party) that the merchant has accepted the authorization response. (This is the payment if Y/N is yes, and the statement of rejection otherwise.) This

is because no other party is capable of finding V. (It would require inverting the one-way function $\mathcal{H}$ on the point $\mathcal{H}(V)$.) Note it is important here that the customer check Sig_A– else an adversary can flip Y/N after the merchant sends Y/N and V. Thus the combination of Sig_M, V and Sig_A give the buyer undeniable proof of the seller's agreement to the transaction. The same could be achieved by M signing A's authorization message, but at the cost of an additional signature.

Obviously, M can refuse forwarding A's authorization message to the customer and sending its last message. In this case, C does not know whether the transaction was aborted or finalised (this must be handled based on the next statement of account).

5.3 3KP

As can be expected, in the last protocol – 3KP – all protocol participants, including customers, possess a public key, with the associated secret key and certificate. As illustrated in Figure 7, all parties are now able to provide non-repudiation.

The CERT_C sent to the merchant may not only contain the customer's public key and ID, but also further data. This further data is included in the certificate *in hashed form* in order to be able to reveal it to the merchant only on demand. For instance, CERT_C might include the hash of the Customer's physical address, and if ordered goods should be sent to C's home address, C can reveal "Customer's physical address" to the merchant who can verify it based on CERT_C.

The customer's signature serves as undeniable proof of transaction (A1.b), and enables disputability (C1.b). On the other hand, the merchants can link all payments of the customer with CERT_C and C's signature, i.e., the customer loses some of the privacy compared to 1KP and 2KP. One way to avoid this is by encrypting CERT_C and the signature with A's public key.

Notice that in 3KP the use of PIN numbers is only for compatibility with the existent infrastructure. Except for that reason, PINs can be safely omitted since the level of authentication provided by the customer signature is significantly superior to that provided by a PIN.

3KP satisfies all the requirements addressed by 2KP as well as:

A1(b): *Undeniable Proof of Transaction Authorization by Customer.* The customer signs the SLIP using a secret key SK_C known to C only.

M2(b): *Proof of Transaction Authorization by Customer.* Based on C's signature, M can verify that SLIP was signed by C. M cannot verify the correctness of the contents of SLIP, especially not of the PIN.

C1(b): *Unauthorized Payment is Impossible.* Follows from A1.b.

6 Summary and Comparison of the Protocols

The *i*KP protocols vary in the degree of both protection and complexity. They proceed in an incremental path towards electronic payment with strong security features with respect to all parties involved. Practically speaking, it is envisaged that 1KP will represent a short-term, interim step towards payment protocols with stronger security guarantees. Thereafter, 2KP and 3KP can be gradually phased in. Table 1 presents a comparison of the *i*KP protocols.

The *i*KP family can fulfill all stated requirements and, in particular, provide non-repudiatable receipts from the acquirer gateway to the merchant/customer, and from the merchant to the customer. In case that the customer also possesses a public-key pair (3KP), non-repudiation becomes possible also from the customer to the merchant/acquirer.

The protocols do not reveal the identity of the customer to the merchant. Order privacy against eavesdroppers could be achieved by applying a secure communication protocol (e.g., SHTTP [18] or SSL [14]), or, if desired, the *i*KP protocols themselves could be extended to provide that protection. Since *i*KP aims at credit-card-like payments, no anonymity against the payment system is provided. Adding anonymity and privacy to all payments is a major change in "payment culture" and only after the deployment of *i*KP-like systems will it be assessable whether the involved parties are inclined to move further into this direction.

The *i*KP protocols can be extended to support batch processing of payments from the same customer by the merchant, or to guarantee amounts as commonly done, for example, in the case of car rentals.

The protection of the acquirer from the Internet is another important aspect to the acceptability of such payment systems - first designs to minimize this exposure have been accomplished.

REQUIREMENTS/PROTOCOLS	1KP	2KP	3KP
Issuer/Acquirer			
A1. Proof of Transaction Authorization by Customer	√	√	√√
A2. Proof of Transaction Authorization by Merchant		√√	√√
Merchant			
M1. Proof of Transaction Authorization by Acquirer	√√	√√	√√
M2. Proof of Transaction Authorization by Customer			√√
Customer			
C1. Unauthorized Payment is Impossible	√	√	√√
C2. Proof of Transaction Authorization by Acquirer	√√	√√	√√
C3. Certification and Authentication of Merchant		√√	√√
C4. Receipt from Merchant		√√	√√

Table 1: Comparison of the *i*KP payment protocols. A requirement marked by √ is satisfied but not disputable, while √√ indicates that the requirement is satisfied based on an undeniable proof, providing non-repudiation and disputability.

References

[1] R. ANDERSON. Why Cryptosystems Fail. *Communications of the ACM 37/11 (1994) 32-41.* <ftp://ftp.cl.cam.ac.uk/users/rja14/wcf.ps.Z>.

[2] M. BELLARE, P. ROGAWAY. Optimal Asymmetric Encryption. *Proceeding of Eurocrypt '94*, May 1994.

[3] J.-P. BOLY *et al.* The ESPRIT Project CAFE - High Security Digital Payment Systems. *ESORICS '94*, LNCS 875, Springer-Verlag, Berlin 1994, 217-230. <http://www.informatik.uni-hildesheim.de/FB4/Projekte/sirene/projects/cafe/index.html>.

[4] H. BÜRK, A. PFITZMANN. Digital Payment Systems Enabling Security and Unobservability. *Computers & Security*, 8/5 (1989), 399-416. <http://www.informatik.uni-hildesheim.de/ FB4/ Projekte/sirene/lit/abstr89.html/#BuePf_89">.

[5] D. CHAUM. Privacy Protected Payments. *SMARTCARD 2000, Proceedings, North-Holland, Amsterdam 1989, 69-93.*

[6] D. CHAUM. Achieving Electronic Privacy. *Scientific American*, August 1992, 96-101.

[7] P. CHENG, J. GARAY, A. HERZBERG, AND H. KRAWCZYK. Design and implementation of modular key management protocol and IP Secure Tunnel on AIX. In *Proc. 5th USENIX UNIX Security Syposium*, Salt Lake City, Utah, June 1995. Also available from ftp site software.watson.ibm.com: /pub/security/mkmp-ipst-usenix.ps.

[8] W. R. CHESWICK, S. M. BELLOVIN. Firewalls and Internet Security: Repelling the Willy Hacker. *Addison Wesley, 1994.*

[9] CYBERCASH. The CyberCash(tm) System - How it Works. <http://www.cybercash.com/cybercash/cyber2.html>.

[10] DIGICASH. About ecash. <http://www.digicash.com/ecash/ecash-home.html>.

[11] S. DUKACH. SNPP: A Simple Network Payment Protocol. *Computer Security Applications Conference,1992.* <ftp://ana.lcs.mit.edu/pub/snpp/ snpp-paper.ps>.

[12] D. K. GIFFORD, L. C. STEWART, A. C. PAYNE, G. W. TREESE. Switches for Open Networks. *IEEE COMPCON*, March 95. <http://www.openmarket.com/about/technical/>.

[13] R. HAUSER, M. STEINER. Generic Function Extension of WWW Browsers. *IBM Research Division, Zurich Research Lab, March 1995.*

[14] K. E. B. HICKMAN. Secure Socket Library. *Netscape Communications Corp., Feb. 9th, 1995* <http://www.mcom.com/info/SSL.html>.

[15] P. JANSON, M. WAIDNER. Electronic Payment over Open Networks. *Zeitschrift der Schweizerischen Informatikorganisationen xxxx (1995) xxxx-xxxx.* <http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/>.

[16] S. H. LOW, N. F. MAXEMCHUK, S. PAUL. Anonymous Credit Cards. *2nd ACM Conference on Computer and Communication Security, Fairfax 1994.* <http://www.research.att.com/index.html#acc>.

[17] C. NEUMAN, G. MEDVINSKY. Requirements for Network Payment: The NetCheque Perspective. *IEEE COMPCON*, March 95. <ftp://prospero.isi.edu/pub/papers/security/netcheque-requirements-compcon95.ps.Z>.

[18] E. RESCORLA, A. SCHIFFMAN. The Secure HyperText Transfer Protocol. Internet Draft. *Enterprise Integration Technologies, December 1994.* <http://www.eit.com/projects/s-http/>.

[19] M. SIRBU, J. D. TYGAR. NetBill: An Internet Commerce System. *IEEE COMPCON, March 95.* <http://www.ini.cmu.edu/netbill/CompCon.html>.

[20] L. H. STEIN, E. A. STEFFERUD, N. S. BORENSTEIN, M. T. ROSE. The Green Commerce Model. *First Virtual Holdings Incorporated, October, 1994.* <http://www.fv.com/tech/ green-model.html>.

A The encryption function

Here we describe the preferred way to get the function $\mathcal{E}_A$ using which the customer encrypts the SLIP under the acquirer's public key PK_A. We will use RSA. We let $f(x) = x^e \pmod N$ denote the RSA function and $f^{-1}(y) = y^d \pmod N$ its inverse, where N is a 1024 bit modulus. The issue is that simply encrypting under RSA– ie. setting $\mathcal{E}(x) = f(x)$– is not enough: this doesn't provide the "integrity" or "plaintext awareness" we need. Instead, we will first "embed" a up to 896-bit plaintext into a 1024 bit string r in a very special way and then compute $f(r)$. The scheme we now describe is a simplification of one in [2]. It makes use, in addition to RSA, of the hash function $\mathcal{H}$, and is provably secure assuming $\mathcal{H}$ behaves like a "random function."

Given a data string DATA, encode it into exactly 896 = 1024-128 bits. (This means pad and include length of original data if necessary.) Then encrypt as follows:

(1) Let $x = 0^{64}$. DATA. (So x has length 960 bits)

(2) Pick a random SALT of length 64 bits.

(3) Let $a = x \oplus H_1(\mathrm{SALT})$ and then let $b = \mathrm{SALT} \oplus H_2(a)$. Here $H_1()$ is a hash function outputting 960 bits and $H_2()$ is a hash function outputting 64 bits. Example ways to compute them starting from given $\mathcal{H}$ are given below.

(4) Let $r = a\,.\,b$ (this is 1024 bits) and output $f(r)$.

It is important to check the redundancy 0^{64} upon decrypting. You recover $r = a.b$, and then compute $\mathrm{SALT} = H_2(a) \oplus b$ and $x = H_1(\mathrm{SALT}) \oplus a$. If x doesn't have 64 leading zeros then reject.

For the hash functions H_1, H_2 one could use:

- $H_2(\mathrm{Text})$ = First 64 bits of output of $\mathcal{H}(\mathrm{Text})$.
- $H_1(\mathrm{Text})$ = First 960 bits of the sequence $\mathcal{H}(0, \mathrm{Text}).\mathcal{H}(1, \mathrm{Text}).\mathcal{H}(2, \mathrm{Text})$....

- **Quantities occuring in all three protocols:**

SALT_C	Random number generated by C; used to salt DESC and thus ensure privacy of DESC on the M to A link
PRICE	Amount and currency
DATE	Merchant's date/time stamp, used for "coarse grained" replay protection of a payment
NONCE_M	Merchant's nonce (counter or random number) used for more "fine grained" replay protection of a payment
ID_M	Merchant id. This identifies merchant to acquirer.
TID_M	Transaction ID. This is an identifier chosen by merchant which uniquely identifies the context
DESC	Description of purchase/goods, and delivery address. Includes payment information such as credit card name, bank identification number, and currency.
CAN	Customer's Account Number (Eg. credit card No.) Includes expiration date.
R_C	Random number chosen by C to form CID.
CID	A customer pseudo-ID which uniquely identifies C; computed as CID = $\mathcal{H}(R_C, \text{CAN})$.
Y/N	Response from the clearing network: YES/NO or authorization code.
Text_j	For $j = 0, 1, 2, \ldots$. This is optional information that can accompany the flows. For example, can be used to carry context identifiers.

- **Quantities occuring in some of the protocols:**

PIN	Customer PIN which, if present, can optionally be used in 1KP to enhance the security
V	Random number generated by merchant in 2KP and 3KP for use as a proof that merchant has accepted payment

Figure 3: *Definitions of atomic fields used in iKP protocols. Composite fields formed from these are discussed later.*

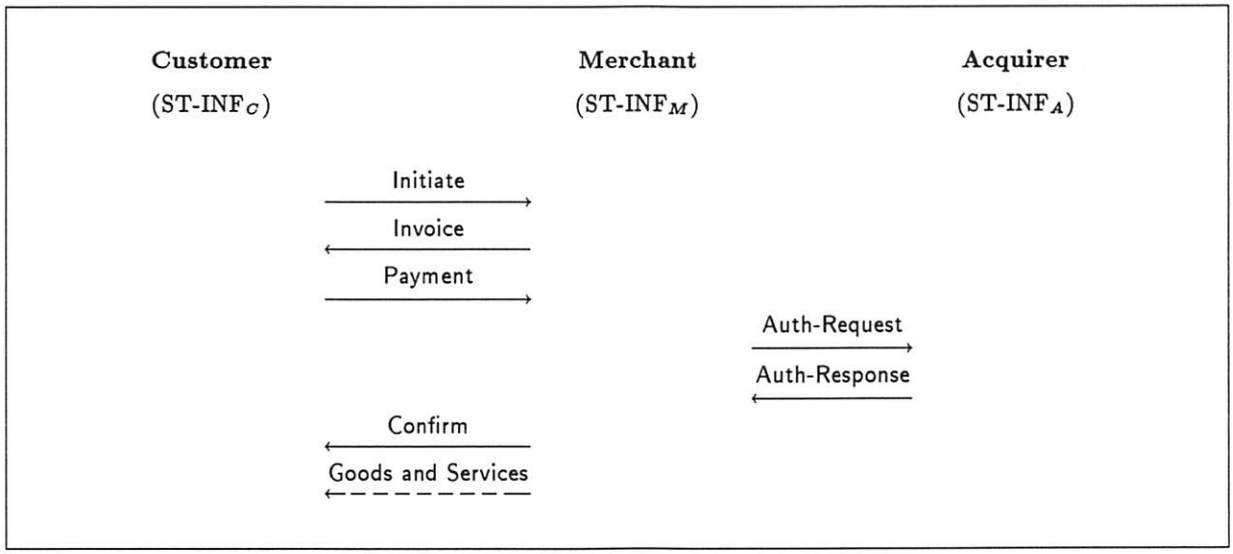


Figure 4: *Framework of iKP protocols*

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}$(Common)
SLIP	PRICE, $\mathcal{H}$(Common), CAN, R_C, [PIN].
EncSlip	$\mathcal{E}_A$(SLIP)

- **Starting information of parties:**

ST-INF_C	$\mathrm{DESC}, \mathrm{CAN}, \mathrm{PK}_{CA}$, [PIN]
ST-INF_M	$\mathrm{DESC}, \mathrm{PK}_{CA}, \mathrm{CERT}_A$
ST-INF_A	$\mathrm{SK}_A, \mathrm{CERT}_A$

- **Protocol Flows:**

Initiate: C $\xrightarrow{\mathrm{SALT}_C,\ \mathrm{CID},\ [\mathrm{Text}_0]}$ M

Invoice: C $\xleftarrow{\overbrace{\mathrm{ID}_M,\ \mathrm{TID}_M,\ \mathrm{DATE},\ \mathrm{NONCE}_M,\ \mathcal{H}(\mathrm{Common})}^{\text{Clear}},\ [\mathrm{Text}_1]}$ M

Payment: C $\xrightarrow{\overbrace{\mathcal{E}_A(\mathrm{SLIP})}^{\text{EncSlip}},\ [\mathrm{Text}_2]}$ M

Auth-Request: M $\xrightarrow{\text{Clear},\ \mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C),\ \text{EncSlip},\ [\mathrm{Text}_3]}$ A

Auth-Response: M $\xleftarrow{\mathrm{Y/N},\ \overbrace{\mathcal{S}_A(\mathrm{Y/N}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_A},\ [\mathrm{Text}_4]}$ A

Confirm: C $\xleftarrow{\mathrm{Y/N},\ \mathrm{Sig}_A,\ [\mathrm{Text}_5]}$ M

Figure 5: 1KP Protocol

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$, $\mathcal{H}(V)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}(V)$, $\mathcal{H}(\mathrm{Common})$
SLIP	PRICE, $\mathcal{H}(\mathrm{Common})$, CAN, R_C.
EncSlip	$\mathcal{E}_A(\mathrm{SLIP})$
Sig_M	$\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))$

- **Starting information of parties:**

$\mathrm{ST\text{-}INF}_C$	$\mathrm{DESC}, \mathrm{CAN}, \mathrm{PK}_{CA}$
$\mathrm{ST\text{-}INF}_M$	$\mathrm{DESC}, \mathrm{PK}_{CA}, \mathrm{CERT}_A, \mathrm{SK}_M, \mathrm{CERT}_M$
$\mathrm{ST\text{-}INF}_A$	$\mathrm{PK}_{CA}, \mathrm{SK}_A, \mathrm{CERT}_A$

- **Protocol Flows:**

Initiate: C $\xrightarrow{\mathrm{SALT}_C,\ \mathrm{CID},\ [\mathrm{Text}_0]}$ M

Invoice: C $\xleftarrow{\overbrace{\mathrm{ID}_M,\ \mathrm{TID}_M,\ \mathrm{DATE},\ \mathrm{NONCE}_M,\ \mathcal{H}(V),\ \mathcal{H}(\mathrm{Common})}^{\mathrm{Clear}},\ \overbrace{\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))}^{\mathrm{Sig}_M},\ \mathrm{CERT}_M,\ [\mathrm{Text}_1]}$ M

Payment: C $\xrightarrow{\overbrace{\mathcal{E}_A(\mathrm{SLIP})}^{\mathrm{EncSlip}},\ [\mathrm{Text}_2]}$ M

Auth-Request: M $\xrightarrow{\mathrm{Clear},\ \mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C),\ \mathrm{EncSlip},\ \mathrm{Sig}_M,\ \mathrm{CERT}_M,\ [\mathrm{Text}_3]}$ A

Auth-Response: M $\xleftarrow{\mathrm{Y/N},\ \overbrace{\mathcal{S}_A(\mathrm{Y/N}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_A},\ [\mathrm{Text}_4]}$ A

Confirm: C $\xleftarrow{\mathrm{Y/N},\ V,\ \mathrm{Sig}_A,\ [\mathrm{Text}_5]}$ M

Figure 6: 2KP Protocol

- **Composite Fields:**

Common	PRICE, ID_M, TID_M, DATE, NONCE_M, CID, $\mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C)$, $\mathcal{H}(V)$
Clear	ID_M, TID_M, DATE, NONCE_M, $\mathcal{H}(V)$, $\mathcal{H}$(Common)
SLIP	PRICE, $\mathcal{H}$(Common), CAN, R_C.
EncSlip	$\mathcal{E}_A$(SLIP)
Sig_M	$\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))$
Sig_C	$\mathcal{S}_C(\mathrm{EncSlip}, \mathcal{H}(\mathrm{Common}))$

- **Starting information of parties:**

$\mathrm{ST\text{-}INF}_C$	$\mathrm{DESC}, \mathrm{CAN}, \mathrm{PK}_{CA}, \mathrm{SK}_C, \mathrm{CERT}_C$
$\mathrm{ST\text{-}INF}_M$	$\mathrm{DESC}, \mathrm{PK}_{CA}, \mathrm{CERT}_A, \mathrm{SK}_M, \mathrm{CERT}_M$
$\mathrm{ST\text{-}INF}_A$	$\mathrm{PK}_{CA}, \mathrm{SK}_A, \mathrm{CERT}_A$

- **Protocol Flows:**

Initiate: C $\xrightarrow{\mathrm{SALT}_C,\ \mathrm{CID},\ \mathrm{CERT}_C,\ [\mathrm{Text}_0]}$ M

Invoice: C $\xleftarrow{\overbrace{\mathrm{ID}_M,\ \mathrm{TID}_M,\ \mathrm{DATE},\ \mathrm{NONCE}_M,\ \mathcal{H}(V),\ \mathcal{H}(\mathrm{Common})}^{\mathrm{Clear}},\ \overbrace{\mathcal{S}_M(\mathcal{H}(\mathrm{Common}), \mathcal{H}(V))}^{\mathrm{Sig}_M},\ [\mathrm{Text}_1]}$ M

Payment: C $\xrightarrow{\overbrace{\mathcal{E}_A(\mathrm{SLIP})}^{\mathrm{EncSlip}},\ \overbrace{\mathcal{S}_C(\mathrm{EncSlip}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_C},\ [\mathrm{Text}_2]}$ M

Auth-Request: M $\xrightarrow{\mathrm{Clear},\ \mathcal{H}(\mathrm{DESC}, \mathrm{SALT}_C),\ \mathrm{EncSlip},\ \mathrm{Sig}_M,\ \mathrm{Sig}_C,\ [\mathrm{Text}_3]}$ A

Auth-Response: M $\xleftarrow{\mathrm{Y/N},\ \overbrace{\mathcal{S}_A(\mathrm{Y/N}, \mathcal{H}(\mathrm{Common}))}^{\mathrm{Sig}_A},\ [\mathrm{Text}_4]}$ A

Confirm: C $\xleftarrow{\mathrm{Y/N},\ V,\ \mathrm{Sig}_A,\ [\mathrm{Text}_5]}$ M

Figure 7: 3KP Protocol

A Set of Protocols for Micropayments in Distributed Systems

(extended abstract)

Lei Tang
Graduate School of Industrial Administration
Carnegie Mellon University
Email: ltang@cs.cmu.edu

Abstract

Micropayments refer to low-value financial transactions ranging from several pennies to a few dollars. A big portion of electronic commerce occurring in the Internet belong to the category of micropayments. The cost of micropayments should be kept as low as possible in order for the service provider (the merchant) to profit from the low-value transactions. We propose several protocols for micropayments in distributed systems. Our main goal is to reduce the charging cost by choosing a suitable security model, a charging model, and cryptographic algorithms; and by employing the properties unique to the information goods. Our protocols satisfy the requirements of a payment system and are "cheap" in terms of computation costs, communication costs, and key management costs. We select the debit model for designing our protocols and base our protocols on the private key cryptosystems. We show that the existing authentication protocols and systems can be extended to handle micropayments in distributed systems. We also present possible extensions to our protocols.

1 Introduction

With the rapid growth of the Internet, more and more computer users rely on computer networks for every kind of information ranging from daily news and journal papers to movies. Most of the information items on the Internet have a low value, ranging from pennies to several dollars. A low-cost electronic charging mechanism should be provided to the intellectual property owners so they can profit from providing these services, and also to stimulate them to provide quality services to the Internet community.

There exists several electronic payment protocols [12, 3]. They are based on different charging models and use different cryptographic algorithms. Security and privacy are main concerns of these protocols. Less is studied about how to address the issue of lowering the transaction cost. We analyze trade-offs of different charging models and different cryptographic algorithms. We also observe that the information goods have two unique properties: they are easy to duplicate without extra cost; and they are unreturnable once purchased. Based on this observation, a set of "cheap" payment protocols is designed. The charging model we use is the debit model. We use only the private key cryptosystems, with the option extended to use the public key cryptosystems. We also give a detailed security analysis of our protocols. Authentication in distributed system has been studied for almost twenty years and a lot of practical systems have been built [17, 18]. Is it possible to extend those systems to process micropayments in distributed systems so that we do not have to build our system from scratch, in turn, the costs of building the system are reduced? we describe the relationship between our protocols and the protocols for authentication in distributed systems and show that it is possible to extend those authentication systems for the purpose of micropayments in distributed systems.

In section 2 we define the parties involved in our payment protocols, and notations used throughout this paper. In section 3 we analyze the elements affecting the transaction cost, the trade-offs among them. Then we choose the on-line debit model to design our protocols. In section 4 we describe the structure of our payment systems. In section 5 we present our protocols, giving a detailed security analysis of these protocols. We also discuss the relationship between our protocols and authentication protocols. We conclude in section 6.

2 Principals and Notations

Every payment system involves at least two parties exchanging goods (or services) and money. We

call the parties involved in the electronic payment system the *principals*. All principals communicate through a computer network. The basic units carrying out the communication in favor of the principals are *processes*. We call the processes used to export services the *servers* and processes used to import services the *clients*. The principal who provides (or sells) services is called the *merchant*, and the principal who imports (or buys) services, the *customer*. The merchant sells his services through the merchant server and the customer buys the services through the customer client. The principal who wants to illegally benefit from the electronic payments by sabotaging or eavesdropping the communication channel is called the *adversary*.

All principals communicate exclusively by passing messages over a network. We assume that the network is not secure. Not only can an adversary mount passive attacks in which the adversary merely observes the messages passing on the network without interfering with them; the adversary can also mount active attacks, performing a variety of processing on the messages passing on the network. These messages can be selectively modified, deleted, delayed, reordered, duplicated, and inserted into the communication at a later point in time; or be allowed to pass through unaffected [19].

We adopt notation common in the literature for authentication protocols, especially the notation used by Abadi and Needham [2]. We will not make a distinction between the principal and the process which the principal creates to accomplish the transaction on behalf of himself. We use the symbols C, M and B to denote the true identities of the principals (the customer, the merchant and the billing service center [1]). T_a represents the timestamp read from principal A's clock. N_a denotes the nonce [15, 16] generated by the principal A. We also use the symbols below to represent the following.

K_a A's public key.
K_{ab} The key shared between A and B.
$\{x\}_K$ x encrypted with key K.
x, y x concatenated with y.
$A \rightarrow B : x$ A sending message x to B.
I_x Item x.
Id_a The pseudonym of A's identity.
P_x Offered price of item x by the merchant.
ET_x Expiration time of the offered price for I_x.

[1] The definition is given in Section 3.3

3 Elements affecting the transaction cost

3.1 System Security

The key for the success of an electronic payment system is *security*. We use security to refer to the most important properties of an electronic payment system: secrecy, authenticity and integrity. Secrecy refers to denial of access to information by unauthorized principals. (e.g., eavesdropper on the network.) Authenticity refers to validating the source of a message; that is, the message was transmitted by a proper identified sender and is not a replay of a previously transmitted message. Integrity refers to assurance that a message was not modified accidentally or deliberately in transmit by replacement, insertion, or deletion. Nonrepudiation of origin is also a desired characteristic for protection against a sender of a message later denying transmission. The electronic payment system should be able to prevent dishonest principals from illegally gaining financial benefits by deviating from the protocols or colluding with other malicious adversaries. The electronic payment system should also be secure against attacks from the malicious adversaries and the dishonest customers (or merchants).

The merchants and the electronic payment service provider will lose revenue due to the breach of the payment system. Finally, the lost revenue is transferred to the honest customers and the cost will increase. Therefore, security is the key for keeping the transaction cost low.

3.2 Public key vs. private key

The public key cryptosystems [6] provide the mechanism for non-repudiation signatures, and are secure against known plaintext attacks [8], which can not be provided by the private key cryptosystems. However, the public key cryptosystems are unsuitable for electronic micropayments for the following reasons.

First, the cost of public key management is very expensive compared with that of private key management. A trusted key certification authority should be established to register and certify all principals who have the public keys in the electronic payment system. The key certification authority also revokes lost or stolen keys. The key certification authority must make sure that revoked keys are known to all principals. Every principal involved in the payment has to check the revocation list whenever a transaction

is processed. Any unavailability of the revocation list or delay in delivery of the revocation list, which is common in the present distributed environment, will make the principals fail to detect attacks from the adversary. However, in the private key cryptosystem, if a customer (or a merchant) loses the secret key shared between him and the electronic payment service provider, he just needs to notify the electronic payment service provider. No further operation is needed for either party.

Second, the public key cryptosystems are computationally intensive since most of the widely used public key cryptosystems (e.g., RSA [14], ElGamal [7]) involve many exponentiation and multiplication operations in an Abelian group with very big cardinality. Usually a public key algorithm (e.g., RSA) is much slower than a private key algorithm (e.g., DES). The performance degrades dramatically when a payment service provider processes these expensive computations for all its customers in one central server.

Third, most of the practical public key cryptographic algorithms are patented. An electronic payment service provider has to pay royalty for using these algorithms. This will increase the transaction cost.

A lack of a public key for the customer and the merchant does introduce some problems. First, it is impossible for customers to inquire about the price of an item from a merchant privately unless both of them share a secret. Second, it is impossible for the merchant to sign a receipt to the customer which is verifiable by a third party. For the first problem, we solve it in another way by using the *pseudonym* concept introduced by Chaum [4]. For the second problem, we provide a weak form of non-repudiation, which we think is sufficient for micropayment. We will elaborate our method in section 5.

Although the public key methods have advantages over the private key methods, they are expensive. It is proper to make distinction between high value transactions and low value transactions. The security mechanism for a transaction worth several pennies should be different from that for a transaction worth several hundred dollars. Therefore, we should use as few public key operations as possible in our protocol design. But our protocol can be easily extended to be based on the public key cryptosystems.

3.3 Charging models

There are several commonly used models available to us to design our protocols for micropayments. They are the *billing (or subscription) model*, the *credit card model*, the *electronic check model*, the *electronic currency model*, and the *debit model*.

In the billing model, every customer registers with different merchants and obtains services (or goods) from the merchants. The transaction cost is low for this model. However, it is very inconvenient to the customers. If a customer wants to purchase goods from one hundred different merchants, he has to open accounts with these merchants and remember one hundred different encryption (decryption) keys. This is a very tedious task for the customers.

For the credit card model, the customer sends his credit card number to the merchant through some secure channel between them. But the high credit card transaction cost makes this model unsuitable for the micropayments.

The electronic check model is also a candidate. The electronic check model depends on a hierarchical banking infrastructure to transfer funds along a path inside the hierarchical structure. If the system is based on the public key cryptosystems, the banks have to provide on-line key revocation servers for their customers whose secret keys are compromised. These servers must be available to all merchants at any time. Any unavailability of these servers will cause the compromised secret keys to be abused by the adversaries who obtain these secret keys. If the electronic check system is based on the private key cryptosystems, a hierarchical accounting structure has to be established so funds can be transferred. Incorporating such an accounting infrastructure into the existing banking system will be expensive. Moreover, a fund transfer operation involves at least three accounting servers if the customer and the merchant do not share a common accounting server. Furthermore, the merchant has to clear the check on-line before he honors the customer's purchasing request. This means that the computer systems and the communication channels along the check clearing path must be reliable all time, which is not the case in a distributed environment.

The electronic currency is not considered since the electronic check model is just a simplified (or special) electronic currency model.

Based on the above observations, we consider the debit model as the model for our electronic payment protocol design. The money debit model is an on-line system. The customer's bank debits the customer's account, transfers the funds to the merchant's bank; the merchant's bank credits the merchant's account. This is the scenario of the debit model. It is not realistic for us to assume that every bank will provide on-

line transfer service to its customers at the present time. Instead, a trusted electronic payment service provider is established to handle all fund transfers between the customers and the merchants. We call it the *billing service center*. Although the billing service center is a security bottleneck and performance bottleneck of the whole system, the simplicity of the debit model structure can reduce the transaction cost significantly.

4 The structure

Our protocol consists of three principals: the billing service center, the customer and the merchant. We assume that the billing service center is honest and is trusted by both the customer and the merchant. The customer and the merchant may be dishonest. The merchant and the customer open accounts and deposit funds in the billing service center.

The billing service center consists of many servers to handle The authentication and authorization of fund transfers between the customer's account and the merchant's account. The protocols for these operations are described in the next section. Besides the fund transfer operations, the billing service center also handles the following functions:

- Account administration for the customer and the merchant. This includes account openings, closures, fund transfers, balance inquiries, account statements, dispute resolutions, etc.
- Key generation and distribution for the customer and the merchant. The billing service center can also certify public keys for certain classes of customers and merchants at an additional fee.
- Administration of the merchant's services and handling complaints from the customers.

There may be more functions needed for the billing service center. They are not covered in this paper. We will only design protocols that transfer funds between the customer and the merchant securely and cheaply.

The billing service center shares a secret key $K_{cb}(K_{mb})$ with the customer (the merchant) and this key is known only to the customer (the merchant) and the billing service center. The billing service center has a certified public key K_b with the corresponding secret key K_b^{-1}.

A customer maintains a positive balance and authorizes charges against this account. When the billing service center receives an authorization by the customer, it checks the validity of the authorization, and debits the customer's account by the amount of the price agreed to by both the customer and the merchant. The billing service center then credits the merchant's account and sends an acknowledgement to the merchant.

Since the customer shares a secret with the billing service center and does not have a certified public key, the billing service center may forge the authorization by a customer, which can not be judged by a third party. This kind of attack is more likely from an insider (e.g., a malicious system administrator of the billing service center) than from an outsider. It can be prevented using the audit method. Moreover, since the amount of the payment is very low, the small cost due to this fraud can be covered by the billing service center, or the customer can change to another electronic payment service provider offering high quality services [2].

5 Our protocols

Before the transaction begins, the customer queries the merchant about the price of specific goods. Since we assume that neither the merchant nor the customer has a certified public key associated with him, it is impossible for both of them to establish a secure channel for the price information. Sometimes it is important to prevent a merchant from knowing the identity of his customers. It is also important to prevent the eavesdropper from knowing what the customer is purchasing and what the price is. We provide a weak form of a secure channel between the customer and the merchant by adopting the *pseudonym* concept [4]. A pseudonym is a nickname of a principal (not his true identity). A principal can have many pseudonyms. When a customer opens an account in the billing service center, several pseudonyms are assigned to the customer and these pseudonyms are known only to the customer and the billing service center. These pseudonyms are used by the customer to inquire about price information and to order goods from the merchants. Since the eavesdroppers and the merchants do not know the mapping between the pseudonyms and the true identity of a customer, the customer's privacy is protected.

This is a weak form of secrecy compared with the

[2] There are many such billing service centers in the Internet. They compete with each other.

1. $C \rightarrow M : Id_c, I_x, P_x?$
2. $M \rightarrow C : \{Id_c, M, I_x, P_x, T_m, ET_x\}_{K_{mb}}, I_x, P_x, T_m, ET_x$

Figure 1: Price Negotiation Protocol

public key approach. But it is more efficient. The price negotiation protocol is shown in Figure 1.

P_x is either a number denoting the price of the item x; or a list denoting the price policy of setting the price for the customer with pseudonym Id_c. For example, P_x can denote the list (*student* \$0.5 *member* \$1.0 *nonmember* \$1.5). In the later case, the customer knows what price he should pay since it knows which class he belongs to, while the adversary can not figure out what the price is for the customer with pseudonym Id_c. This is also a weak form of privacy for the price information. The price policy is enforced by the billing service center since it knows which category the customer belongs to. We call the message $\{Id_c, M, I_x, P_x, T_m, ET_x\}_{K_{mb}}$ the *price-form*. In case the merchant has a public key, then the price-form can be signed by his secret key K_m^{-1}. We may send a checksum on (I_x, P_x, T_m, ET_x) to prevent the tuple from being modified by the adversaries during its transmission. Even without the checksum, any modification on the tuple will be detected by the billing service center and the adversaries gain nothing from their malicious attacks.

Based on different communication paths among these three parties shown in Fig 2. we propose three protocols.

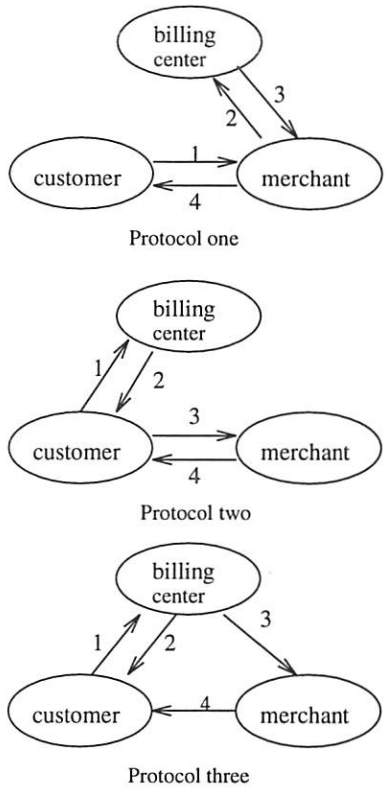


Figure 2: Design alternatives

1. $C \rightarrow M : orderform$
2. $M \rightarrow B : orderform, Id_c, M, \{Id_c, M, K_{cm}, T'_m\}_{K_{mb}}$
3. $B \rightarrow M : \{ok, Id_c, M, I_x, P_x, T_b, \{C, M, K_{cm}, T_b\}_{K_{cb}}\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}, \{C, M, K_{cm}, T_b\}_{K_{cb}}$
   if ok from B.

Figure 3: Protocol one

5.1 Protocol one

Before the transaction begins, the customer obtains the price-form from the merchant through the price negotiation protocol. Our first protocol includes the following steps: The customer sends his payment authorization together with the price-form to the merchant; the merchant forwards this payment authorization to the billing service center. The billing service center checks the validity of the authorization, debits the customer's account by the amount specified in the authorization and credits the merchant's account. After the merchant receives an acknowledgement from the billing service center, the merchant provides the goods (or services) to the customer. The billing service center also generates a random key used by the customer and the merchant to encrypt the communication between them. In this protocol, the merchant communicates with both the customer and the billing service center. There is no communication between the customer and the merchant.

After running the protocol, C and M share key K_{cm}, C can then decrypt the information sent to him by M; or M rejects C's order since he does not have enough funds in his account or there is an inconsistency between the order-form and the price-form. We call the token $\{C, M, I_x, P_x, ET_x, T_c, serialnum, priceform\}_{K_{cb}}$ the *order-form*. The protocol is explained step by step:

1. The customer creates a fund transfer authorization containing

   –the merchant's and the customer's identities;

   –the item x and the associated price P_x;

   –a timestamp T_c read from the customer's clock;

   –the serial number for the order; and

–the price-form signed by the merchant during the price negotiation procedure.

Double encryptions are used to guarantee that P_x is agreed upon by both C and M. The merchant can not change it after it is signed. This is the order-form from C to M and is sent to B later for verification. A timestamp is used to guarantee the freshness of the price-form and the order-form [5]. The *serialnum* serves two functions here: one is to serve as a "confounder" to stop password guessing attacks from malicious merchants or eavesdroppers [11]; another one is to make it easy for the customers to record their purchases (like the serial number in personal checks).

2. M sends to the billing service center the payment agreement between C and M signed by both parties. K_{cm} is the decryption key of C which is used to decrypt the encrypted information sent to C by M. It is encrypted together with a timestamp by K_{mb} to guarantee the secrecy and freshness of K_{cm}. K_{cm} will be recorded with other records of this transaction by B. When B receives the authorization from C, it extracts the order-form and the price-form using the secret key K_{cb}. Then it checks if the price-form is consistent with the order-form. This includes checking whether the prices are the same, and whether the price offer is still in effect. Moreover, it verifies the timeliness of the order-form. If the message is determined to be valid, B debits the customer's account by the amount specified in P_x and credits the merchant's account. In case P_x is a price list, B sets the price for the merchant according to the price policy. If the customer has not enough funds in his account or the customer is not eligible to purchase the goods (e.g., a customer under eighteen years old is not allow to buy pornographic materials), B acknowledges to C and the transaction aborts. Otherwise, B goes to the next step.

3. B acknowledges to M that he has transferred funds to the merchant's account by including the following fields in the acknowledgement form: the merchant's identity and the customer's pseudonym; the item and the corresponding price; the timestamp and the encrypted secure communication key between C and M. Double encryption here is to prevent C from getting $\{C, M, K_{cm}, T_b\}_{K_{cb}}$ and obtaining the key K_{cm} directly from B. It guarantees that C obtains the decryption key from M directly. If this is not a requirement, then the double encryption is not necessary here. Since we include M inside $\{C, M, K_{cm}, T_b\}_{K_{cb}}$, M's identity is also authenticated to the customer.

4. M sends the decryption key to C. The encryption file can be sent to C by M at any step during the protocol.

Usually, at the end of transaction, the merchant should give the customer a receipt. The receipt serves as a function that the customer can prove to the merchant or a third party that the transaction between the merchant and him did happen. In most of the cases the receipt is used by the customer to return goods to the merchant or to present the receipt to a third party for refund. But information goods have the special property that they can not be returned once purchased. Since the merchant does not have a public key, it is impossible for him to sign a verifiable receipt to the customer. Under our structure, the merchant can provide the customer a weak form of receipt. Because all transactions go through the billing service center, the receipt can be provided by the billing service center to the customer if necessary. The receipt has the form $\{Id_c, M, I_x, P_x, ET_x, T_c\}_{K_b^{-1}}$. The billing service center can also serve as a "witness" if a dispute occurs between the customer and the merchant. In this case, the billing service center does not have to sign the receipt.

At the final step of the protocol, the merchant sends $\{content(I_x)\}_{K_{cm}}$ to the customer. A dishonest customer may claim to the billing service center that he has not received the goods even through he did. The billing service center can ask the merchant to send the goods again. Unlike physical goods, information goods are easy to replicate, and the merchant can send duplication of a given good many times.

Neither the customer nor the merchant can modify the price in the price-form or the order-form during the transaction, since otherwise the billing service center can detect the inconsistency and abort the transaction.

If a dishonest merchant sells low-quality goods to the customer, the customer can complain to the billing service center. The billing service center who administrates all merchants can stop providing service to those merchants with poor reputations.

A dishonest merchant may collude with other dishonest merchants or dishonest customers by showing them the order-form from his customers. Since the merchant's identity and a timestamp are included in

1. $C \rightarrow M : \{C, M, I_x, P_x, ET_x, \{C, serialnum, T_c\}_{K_{cb}}\}_{K_b}$
2. $M \rightarrow B : \{C, M, I_x, P_x, ET_x, \{C, serialnum, T_c\}_{K_{cb}}\}_{K_b}$, $\{M, Id_c, I_x, P_x, ET_x, \{Id_c, M, K_{cm}, T_m\}_{K_{mb}}\}_{K_b}$
3. $B \rightarrow M : \{\{ok, Id_c, M, I_x, P_x, T_b, \{C, M, K_{cm}, T_b\}_{K_{cb}}\}_{K_b^{-1}}\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}, \{C, M, K_{cm}, T_b\}_{K_{cb}}$

Figure 4: Protocol one based on public key method

the order-form, the order-form can not be reused by other merchants.

The biggest threat from the adversary is the key guessing attack, since the format of the price-form and the order-form are predetermined. Although the exact value of T and *serialnum* may not be known in advance, a plausible range of values can be determined. This structure property generally increases the vulnerability of these messages to a guessing attack. The vulnerability is worsened by the generation of the keys such as K_{cb} and K_{mb}. Usually, a customer of the billing service center (here the merchant is a customer of the billing service center) chooses a password or a PIN number. This password or pin number, together with a random number, are applied to a one-way hash function, and thus the secret keys shared between the customer and the billing service center are generated. The structure properties and the way that the keys are generated make the protocol weak against password-guessing attacks since all the attacks can be done off-line without being perceived. For example, an eavesdropper can collect the order-forms from the customer C and take a guess of the key K'_{cb}. If $\{orderform\}_{K'_{cb}}$ contains C and M, this means that his guess is right with high probability. To prevent this kind of attack, we extend our protocol based on the public key cryptosystem. (only the billing service center has a public key.) The public key cryptosystem is known to be secure against the known plain-text attacks [8]. The modified protocol is shown in Figure 4.

5.2 Protocol two and protocol three

Before the transaction begins, the customer obtains the price-form from the merchant through the price negotiation procedure. Protocol two is shown in Fig 5. The protocol consists of the following steps: In message 1, the customer sends the price-form from and his order-form to B directly. B checks the consistency between the order-form and the price-form which are signed by C and M respectively. If they are consistent, B debits C's account and credits M's account, and generates a key K_{cm} for secure communication between C and M. He then sends the key to C in message 2, and sends the acknowledgement form $\{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$ to M. In message 3, C forwards the acknowledgement form to M so that M can check the validity of the form. M then sends the content of I_x to C using the encryption key K_{cm} in message 4.

1. $C \rightarrow B : \{M, Id_c, I_x, P_x, ET_x, T_m\}_{K_{mb}}$, $\{C, M, I_x, P_x, ET_x, T_c\}_{K_{cb}}$, Id_c, M
2. $B \rightarrow C : \{C, M, I_x, P_x, K_{cm}, T_b, balance\}_{K_{cb}}$, $\{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$
3. $C \rightarrow M : \{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$, $\{Id_c, addr, T_c\}_{K_{cm}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}$

Figure 5: Protocol two

1. $C \rightarrow B : \{M, Id_c, I_x, P_x, ET_x, T_m\}_{K_{mb}}$, $\{C, M, I_x, P_x, ET_x, T_c\}_{K_{cb}}$, Id_c, M
2. $B \rightarrow C : \{C, M, I_x, P_x, K_{cm}, T_b, balance\}_{K_{cb}}$
3. $B \rightarrow M : \{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}$

Figure 6: Protocol three

The only difference between protocol three and protocol two is that B sends the acknowledgement to the merchant directly without going through the customer. Protocol three is shown in Fig 6. Security analysis for these two protocols is exactly the same as that we did for protocol one.

5.3 Discussion

Our protocols are quite similar to those protocols designed for authentication in distributed systems. If the price policy set by the merchant is that the price is zero for every customer in a specific group, our payment protocols can function as authentication protocols (to authenticate if a customer belongs to a group). In this case, protocol two is in the style of the protocol designed by Needham and Schroeder [15] and is the same as the Kerberos protocol [17]. Protocol one is in the style of the protocol designed by Otway and Rees [13]. Our electronic payment protocols can be regarded as extensions of the protocols for authentication in distributed systems. The formal method [1] can also be applied to our protocols. Some practical authentication system such as Kerberos can be extended to handle micropayments if we add additional fields into the Kerberos message structure. This means that we do not have to build an electronic payment system from scratch, which will save us a lot of work for implementation and therefore reduce the transaction costs.

Protocol one is suitable for an electronic payment system in which the merchants are put together in the same realm. Protocol two is suitable for an electronic payment system in which the customers are in the same realm.

The "cheapness" of our protocols is achieved in the following aspects.

- *Communication complexity.* The number of messages passing among the customer, the merchant, and the billing service center is optimal [9, 10].
- *Computation complexity and key management complexity.* We base our protocols on the private key cryptosystems. Our protocol requires significantly less computation than its public key counterparts. The disadvantages introduced by using the private key cryptosystems are remedied by employing the properties unique to the information goods. The secrecy of the price negotiation between the customer and the merchant is guaranteed by the pseudonym concept, which can be implemented cheaply. The receipt of the purchase is achieved in a cheap way due to the following reasons. First, the information goods can not be returned once purchased. Second, the information goods are easy to copy. The merchant can send the information goods to a customer several times if the customer claims that he has not received the information goods which has been paid.
- *Simple banking structure.* We use a debit model consisting of three parties. This model reduces the interactions with the existing banking system, hence also reduces the transaction costs.

6 Conclusions

We have presented a set of protocols suitable for micropayments in distributed systems. Our protocols are "cheap" and satisfy the requirements that a payment system should have. We reduce the charging cost in two ways. First, we analyze the tradeoffs among different charging models and different cryptographic algorithms and base our choice on this analysis. Second, we observe the special properties unique to the information goods and optimize our protocols based on these observations. Our protocols are also natural extensions to the protocols for authentication in distributed systems. The scalability of the system based on our protocol suffers if the system grows. Further study is needed to address this problem.

7 Acknowledgements

I am grateful to Robert Carr for his helps to improve the presentation of this paper. I also want to thank the following people for their comments and constructive criticisms: Li Gong (SRI International), David Kristol (AT&T Bell Labs), and Bennet Yee (Microsoft).

References

[1] M. Abadi, M. Burrows, and R. Needham. A Logic of Authentication. *ACM Transaction on Computing Systems*, 8(1):18–36, 1990.

[2] M. Abadi and R.M. Needham. Prudent Engineering Practice for Cryptographic Protocols. Technical Report SRC 125, DEC System Research Center, 1994.

[3] M. Bellare and et al. iKP–A Family Secure Electronic Payment Protocols. Technical report, IBM Zurich Research Lab, 1995.

[4] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. of ACM*, 24(2):84–88, 1981.

[5] D.E.Denning and G.M.Sacco. Timestamps in Key Distribution Protocols. *Communications of ACM*, 24(8), 1981.

[6] W. Diffie and M.E. Hellman. New directions in Cryptography. *IEEE Transaction on Information Theory*, 22(6):644–654, 1976.

[7] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. Information Theory*, IT-31(4):469–472, 1985.

[8] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive-Chosen Message Attacks. *SIAM J. of Computing*, 17(2), 1988.

[9] Li Gong. Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *Proceedings of the 1st ACM conference on Computer and Communication Security*, Nov 1993.

[10] Li Gong. Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations. Technical report, SRI International, Computer Science Lab, 1994.

[11] L.Gong, M. Lomas, R.M. Needham, and J.H. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communication*, 11(5), 1993.

[12] C. Neuman and G. Medvinsky. Requirements for Network Payment: The Netcheque Perspective. In *Proceedings of IEEE Compcon'95*, March 1995.

[13] D. Otway and O. Rees. Efficient and Timely Mutual Authentication. *Operating System Review*, 21(1), 1987.

[14] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of ACM*, 21(2), 1978.

[15] R.M.Needham and M.D.Schroeder. Using encryption for Authentication in Large Networks of Computers. *Communications of ACM*, 21(12), 1978.

[16] R.M.Needham and M.D.Schroeder. Authentication Revisted. *Operating System Review*, 21(1), 1987.

[17] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the USENIX Winter Conference*, pages 191–202, February 1988.

[18] J. Tardo and K. Alagappan. SPX: Global Authentication using public key certificates. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 232–244, 1992.

[19] V.L. Voydock and S.T. Kent. Security mechanisms in High-level Network Protocols. *ACM Computing Survey*, 15(2):135–171, June 1983.

The Millicent protocols for electronic commerce

Mark S. Manasse
Systems Research Center
Digital Equipment Corporation
Palo Alto, California
msm@pa.dec.com
http://www.research.digital.com/SRC/people/Mark_Manasse/bio.html

Introduction

Many protocols have been proposed in the last year which address the problem of securely transferring money over a public network. Most of these schemes have similar goals: to enable transactions with properties similar to those achievable today using credit cards, to provide consumers and vendors with guarantees similar to those afforded by credit cards, or to translate existing payment mechanisms into electronic equivalents. The schemes vary widely in details; some schemes create electronic cash, some schemes work hard to provide anonymity for purchasers, some schemes protect consumers from exposing their underlying accounts to merchants, and some schemes ignore privacy issues altogether.

This is all interesting and necessary, but it fails to address the problems of a market segment that I expect to grow rapidly, once it becomes at all possible: extremely low-priced walk-up transactions. The proposed payment schemes all come with fee schedules that limit transactions to be fairly valuable. In practical terms, for today's systems, and today's mechanisms for transactions, the fees come to a minimum of approximately twenty-five cents for credit-card-like transactions, and at least a penny for services that provide a level of aggregation before charging a credit card. Over time these fees are almost certain to drop, but the current performance of disks, processors, and networks suggest that these fees aren't going to come down due to economic pressures alone: a system that can only handle a dozen transactions per second per computer needs to charge fees in this range in order to be profitable—and that's the kind of transaction rate implied by systems that require non-repudiatable digital signatures. Increased performance over time will increase the transaction rate; I contend that it's interesting to look at the results of applying that to decreasing the price of transactions.

With fees in their current range, the minimum transaction is likely to remain between five cents and one dollar. For tangible goods, this is unlikely to be a problem; even penny candy costs a few cents these days! For information goods, however, the costs for manufacturing and delivery are quite small. If the overhead for billing could be reduced to the same level, what changes might take place in information delivery? Does it matter if the smallest transaction is five cents or a mil?

I contend that it does matter. At five cents, one can decompose a daily newspaper into a few sections (sports, business, world news, local news) without significantly increasing the daily outlay for information. But this level of decomposition doesn't change the nature of information delivery. At a mil, we can decompose the newspaper into individual articles, comic strips, and horoscopes.

And this information marketplace isn't limited to newspapers: individual stock quotes, background stories culled from the morgue, technical papers, historical commercial information, index retrieval, almost anything where the information production can either be automated, or where the market is large enough. An anarchic information provision world, where payment can be rendered not only for the information, but for the filtering to provide just the information that's fit to download.

A decade hence, assuming that computers (and their components) continue on the price and performance curves of the last two decades, the minimum transaction grain will be an order of magnitude smaller than it is today. The smallest transactions will be in the

sub-penny range. In a decade, we will see pricing by the web page. We can unbundle the newspaper and the magazine, derive revenue from the newspaper morgue, and sell information without restricting consumers to a subscription to a small number of sources. Citation and reference by hyperlink can replace quotation. "Fair use" photocopying can be replaced by links, and the information gets to the students that care at lower total cost, and with revenue returned to the author rather than the paper and toner companies. Editors, authors, and publishers become more independent, as an editor becomes anyone who can assemble a useful list of pointers to interesting documents, and find a market for that list.

This, I argue, is inevitable. The alternative is not that information remain expensive and bundled, but that copyright become meaningless. Digital storage of information is cheaper than the paper it replaces, and your published works will be photocopied and scanned—as long as it remains overwhelmingly cost-effective to do so.

I have a clear preference between these alternatives. Authors and editors need to be compensated in order to produce quality work. A scheme in which compensation continues to flow to the creators of a piece of work is superior to one in which compensation is incommensurate with use. The remaining question is whether the technology and the will to preserve copyright will leap ahead of the technology and the will to subvert it, or not. I can't speak to matters of resolve, but I will speak to matters of technology: storage, scanning, and OCR technologies are running ahead of the pricing models for current schemes of electronic commerce. While electronic commerce will eventually realize the kinds of economies necessary to preserve copyright, it's not certain that it won't already be too late.

Therefore, it becomes interesting to look at techniques for accelerating the move to sub-penny transactions. There are a few approaches that can be followed.

First, we can look to aggregated service providers. CompuServe, America Online, and (soon) the Microsoft Network provide simple billing models for access to information. The main problem is that the information that they can pay for is information for which they have contracted, and information that they deliver. This makes it hard for smaller information providers to gain access to users, makes it hard for consumers to retain much privacy from their service provider, makes it hard for information providers to determine whether the level of use reported by the service provider is accurate, and makes it hard for consumers to tell whether they are being accurately billed. Do consumers and vendors care about the privacy and accuracy issues? Not so much, today, but a good example of inappropriate behavior by an aggregated service provider might make people care.

Or, we can look at Millicent for an alternative architecture that provides better accuracy guarantees and modest privacy guarantees, if you believe that your payment intermediary isn't snooping all of your network traffic.

Millicent overview

To achieve privacy, control over expenses, guarantees of payment, and lots of other really good properties, it's hard to beat cash. A good cheap scheme for digital cash would solve all of the problems right away, except for the problems caused by anonymous transactions, which any FBI or NSA agent will be happy to explain to you.

The problem is, there is no good cheap scheme for digital cash. Any scheme for digital cash is going to boil down to a signed message from a bank that claims to be worth some amount. Checking the signature is going to be relatively easy (although not immediate), but that's insufficient to tell whether a piece of digital cash is valid. Unlike real cash, duplicating digital cash is cheap and easy; you just send the number a second time to a different merchant. The bank can protect against double spending if the double-spender can be detected and assessed for the amount of duplication; this isn't going to be enough to prevent largely anonymous users from spending the same electronic coin several thousand times, or from skipping out on their debts. In such cases, the bank would have to deny payment to merchants, or make good on such fraud, driving the transaction price up. To protect against this, merchants could check each coin with the bank, to make sure that it hasn't been previously spent. This is going to add latency (due to queueing delays) and expense to a transaction system.

One way to make duplicate checking cheaper is to allow everyone to do it. If a local check on a serial number sufficed, we wouldn't have added much latency to the system. But we can't do this with cash; the idea of cash is that it's universal, and thus spendable at multiple vendors simultaneously. The first thing we'll do in Millicent is to change this.

Let's define a new kind of currency, which we'll call *scrip*. Scrip is like cash, in that it has some intrinsic value, but different in that it has that value only when

spent with a specific merchant. Like electronic cash, scrip will consist of a signed message attesting that a particular serial number holds a particular value. Unlike typical cash, the message will contain an expiration date, and a particular vendor with whom the scrip can be redeemed.

Since we're concerned with providing transactions in the small, we'll allow ourselves some other simplifications. Ordinarily, digital cash is signed in some way that permits easy verification of the authenticity of a coin. Since a piece of scrip is of value only to its creator, and since the value of a piece of scrip is small, we don't need to be able to prove to anyone other than that vendor that a piece of scrip is authentic. We can assume that vendors are, in some way to be described later, controlled, and that consumers don't buy scrip from vendors that have cheated them before; thus, a consumer's risk in holding scrip from a vendor is limited to the face value of that scrip.

The best analogies to scrip are transit-system fare cards, pre-paid phone cards, and manufacturer coupons. A piece of scrip represents pre-paid value, whose authenticity is of interest only to the vendor, as long as the consumer is confident that it came from a trustworthy source. The vendor can employ whatever technology is appropriate to ensure authenticity and non-duplication.

In the scheme we propose for Millicent, the authenticity is guaranteed by a secret key. Since we don't need to prove the value to anyone other than the single merchant, it suffices to compute a signature incorporating a special secret. We propose that computing MD-5 of the plain text of the scrip followed by a secret key is sufficient, although any one-way hash function will suffice. Encryption would also suffice, but would be slower and might give us more trouble exporting the system.

To avoid duplication, the vendor will keep bit vectors corresponding to subranges of the issued serial numbers for scrip. As scrip expires (or becomes fully spent) the bit vectors can be discarded. This allows us to limit the storage requirements for a piece of scrip to an amortized $1+\varepsilon$ bits. This permits the vendor to keep the database of valid scrip in main memory, speeding up transactions.

This leaves us with one big problem: buying scrip for a particular vendor. We can't buy it directly from the vendor, because that would commit us to large accounts with vendors, tying consumers down to large information providers. To get around this, we postulate scrip brokers: agents who buy contracts from vendors to produce scrip.

Millicent scrip

Scrip in Millicent has to be unforgeable, unusable if copied, and inexpensive to create and validate. By using message digest functions as our primary form of encryption, and using shared-key cryptography when cryptography is necessary, we attempt to satisfy these requirements. A piece of scrip in Millicent consists of several parts. The scrip must identify the vendor, the expiration date, the value, and a small amount of information about the consumer. This information is intended to be limited to just the information necessary to satisfy requirements of age, location of residence (for sales tax and tariff purposes), and information needed to qualify for certain discounts, such as student status. Even so, some of this information may be sensitive for certain consumers; thus, we want to make sure that any portion of the data that the user wants to transmit only in encrypted form can be so transmitted.

For administrative purposes, and purposes of making change easily, it is desirable to create session keys. For this purpose, the first two parts of a piece of scrip are a session id and a session key. The session id serves to remind the vendor how to compute the session key. We propose that the session id contain in it the expiration date and a unique session number; if the vendor has a mapping from ranges of session numbers to secrets, the session key is just the result of signing the session id using the corresponding secret.

Let's introduce some notation. H will be a message-digest function, probably MD-5. SessionID will be the string corresponding to the session id; it includes N, the session number. The vendor remembers secrets X(N), where the mapping depends only on, say, the high-order bits of N, and remembers them as long as the expiration period for all sessions that have those high-order bits lasts.

Then the session key, k, is just H(SessionID^X(N)), where ^ denotes concatenation; this is what we mean by "signing a string (in this case, SessionID) with X(N)".

The remaining session information contains the vendor and consumer information. This information we denote by Info; when transmitting Info, we allow the consumer and vendor to decompose Info into encrypted and unencrypted pieces however it suits them, so long as Info can be recovered intact. When encrypting, the encryption should use k as the key.

Finally, we have to encode a coin. A coin, Coin, is a string containing a value, and a serial number Z. Additionally, we compute a coin key, j, a signature witnessing that Coin belongs to the current session. This signature is the result of computing H(SessionID^Info^Coin^X(Z)). For coins minted by a broker, we require that Z=N; we strongly recommend that X(Z) equal X(N) only when Z=N. That is, the same secret applies to a range of Z or N; we recommend that Z and N not be chosen in the same range except when equal, and that vendors choose distinct secrets for distinct ranges.

When a consumer wants to spend a piece of scrip, she sends SessionID, Info, Coin, and j to the vendor together with a request Req, and signs SessionID^Info^Coin^j^Req with k. Note that Info and Req may be sent partially encrypted under k; Req may contain instructions requesting that parts of the reply be encrypted. If any change is due to the consumer, the vendor sends new values for Coin and j back to the consumer along with the results of the request; to allow the consumer to check authenticity of the response, the whole response should be signed with k. Note that the new value for Coin will have a fresh serial number, and a (typically) decremented value.

Since H is a cryptographically strong message digest function, no one can create valid scrip without knowing X(Z) and X(N). In particular, no one can determine a session key without X(N), and no consumer can successfully find the matching value for a coin key after modifying SessionID, Info, or Coin. Thus, only parties knowing X(Z) (i.e., the vendor and broker licensed to create scrip for a subrange) can set Info and Coin; no one other than the vendor can change the monetary value of a piece of scrip or tamper with the consumer information. By choosing X(Z) different from X(N), the vendor prevents the broker from upgrading the value of consumer-held scrip; the vendor can now verify that all scrip for which X(Z) is a shared secret has Z=N, and the coin contains the value agreed to in the contract between the vendor and the broker.

Vendors may want to encrypt portions of their replies to requests, to prevent announcing their proprietary data to the world. For data which is "nearly public", that is, for data where the consumer has not requested privacy, it may suffice to encrypt the data with a single key for all users, and sell consumers that key encrypted by the session key.

We propose that for transmission of keys, it may be more efficient, and sufficiently secure, to use H and k as a pseudo one time pad. That is, by choosing and transmitting a random string, t, H(t^k) can be exclusive or-ed with a piece of data of the same length as the result in order to securely restrict transmission of that data to parties that know k. We particularly see an application of this to brokers, who need to transmit values for SessionID, Info, Coin, j, and k to consumers who have just bought scrip; of these, only k and some pieces of Info need to travel securely. The secure pieces of Info can be encrypted under k; the transmission of k must be accomplished securely, or else anyone observing that transmission could spend the scrip. Our current prototype implementation uses the one-time pad technique for encrypting k.

Validation of scrip is straightforward: SessionID is used to recover k, by signing SessionID with X(N); since N is part of SessionID, this can be found. Info, if partially encrypted, is reconstructed using k, and SessionID, Info, Coin, Req and j are signed using k and matched against the transmitted value. A match proves that the sender either knew k, or retransmitted an old request, which would therefore be using a previously spent serial number. Additionally, SessionID, Info, and Coin are checked for tampering by comparing j to the signature using X(Z) of SessionID, Info, and Coin. Finally, Z is checked against the bit vector for freshness; if it is not fresh, this is either a replay, or an attempt by the consumer to respend used scrip.

Money

To make the system work, actual money needs to change hands sometimes, and real signatures are needed to insure that these transfers are authorized and correspond to desired consequences.

To buy vendor scrip from a broker, the consumer needs some broker scrip. The consumer buys broker scrip by using some higher-priced form of electronic commerce (DigiCash, NetBill, the Visa/Microsoft protocol, SSL, whatever), which hopefully provides the ability to transmit k securely. After that, the consumer trades broker scrip for vendor scrip as needed, typically receiving lots of change, and a smaller quantity of vendor scrip. As mentioned above, this vendor scrip will have its own values for SessionID, Info, Coin, j, and k; at least k must be transmitted securely. We anticipate that consumers (or their browsers) will tune the policies of their software to buy enough scrip to cover some modest but anticipated number of transactions with the vendor; quantity discounts from the broker (or high markup on small purchases) will encourage scrip purchasers in this direction. This helps limit the transaction rate seen by brokers to an

acceptable level. We anticipate that typical purchases will be in the range of five to ten dollars when buying broker scrip, and approximately one cent when buying vendor scrip.

In order to sell vendor scrip, the broker needs to acquire a license to produce scrip. The license needs to be enforceable through normal business practices; we expect that both pre-paid and consignment sale of scrip licenses will occur, with the choice being made by the broker based on past sales volumes and experience. We expect the development of a standard license for small-time information vendors; we anticipate that this will be a consignment sale, and that there will be a conventional discount to brokers for this scrip.

A license consists of a range of N, the secret X(N) for that range, an expiration period, an agreement as to what information Info should contain, and either a fixed value for each piece of scrip in the range or a maximum total value for all broker-produced scrip in the range and a value bound for individual pieces of scrip. As the system scales, it may become necessary for brokers to refer consumers to other brokers to satisfy scrip requests; in that case, the consumer's broker may offer to establish a small account with the vendor's broker for purposes of completing this transaction.

Rekeying, refreshing, and redemption

In order to protect consumer information requests from the broker, consumers may want to rekey their scrip from a vendor using public-key technology. Vendors can be expected to accommodate this request, at a small price; all that's involved is choosing a new value for N, in a range not licensed to any broker, recomputing j and k, and transmitting the new SessionID, Info, Coin, j, and k, with the new k traveling encrypted. The encryption can either be an exclusive-or against a random string provided to the vendor under the vendor's public key, or encryption under a public key specified by the user. Unfortunately, this last is susceptible to a man-in-the-middle attack by the broker; since the only reason to rekey is to divorce your requests from observation by the broker, this may be inadequate, depending on the threat model we ascribe to our broker. The former has the disadvantage that it places the heavier computational load on the vendor, which doesn't scale as well. However, it is immune to man-in-the-middle attack.

Refreshing is similar, except that there's no requirement that the new k be strongly encrypted; the total value of the remaining scrip is decreasing, so there's less reason to be concerned about chaining keys than is typical in cryptographic protocols. We anticipate that refreshing soon-to-expire scrip will be a low-cost operation; vendors may, in fact, prefer to garbage collect subranges to eliminate the storage of bit vectors, and thus may be willing to refresh for free.

At some point, a browser may determine that certain scrip is unlikely to ever be of use. At this time, the consumer might like to redeem the vendor scrip for broker scrip. As part of a broker's license for vendor scrip, vendors receive a small license for generating broker scrip in order to redeem unused scrip. This scrip can either be spent by the consumer in the normal way, or merged with the consumer's existing broker scrip. It's likely that there will be a stiffer fee for these actions, to limit traffic and encourage careful heuristic design for scrip purchase and return.

Problems

We've seen that consumers can't easily cheat in this system: if H is strong, consumers can't create scrip, they can't alter scrip, and they can't respend scrip. But who's to blame when something does go wrong? Vendors would be well-advised to anticipate that a certain fraction of transactions will get dropped at an awkward moment; the consumer's computer or network connectivity will fail, and the consumer both won't get what she wanted, and won't receive the change owed to her. To ameliorate this, vendors should keep an in-memory log of hash values for recent transactions, and generally allow replay of any transaction, or at least be willing to return the change from that transaction if the information previously delivered is now uselessly stale, as long as the request exactly matches the previous request.

If a consumer apparently tries to respend scrip, there are a few possible reasons:

1) the consumer could be cheating—this is what we're trying to catch.

2) the consumer could be confused—by permitting replay, we solve some of this; by writing browsers that are careful, we avoid a lot of this.

3) the broker could be selling the consumer used or bogus scrip—this is a problem that needs to be monitored, making sure that the vendor keeps track of whether clients of particular brokers have more trouble than others. Consumers who have done nothing wrong should insist that their broker make good on rejected scrip; if the broker won't, find another broker.

4) the vendor could be cheating by reporting valid, unspent scrip as bogus or spent—this is the flip side of problem 3, and one that brokers need to be on the lookout for. We anticipate that most brokers will refund scrip to the consumer, until they develop a suspicion of either the vendor or the consumer. If the former, the broker can warn customers wanting vendor scrip of apparent problems with the vendor, demand better licensing terms, or stop doing business with the vendor. If the latter, the broker stops doing business with the customer. Consumers can't tell the difference between 3 and 4, except that their broker will be more helpful in case 4.

5) the cryptographic system in use could be insufficiently strong, or the security of the computers involved in producing or holding secrets may be inadequate. Stay vigilant.

Performance

In the scheme outlined above, if no data is encrypted, the cost of a normal purchase is 3 hash functions for validation, and 1 to make change, plus the table lookups. Buying scrip from a broker is similar, plus the scrip generation costs: 2 hash functions to produce j and k, and one more using the suggested one-time pad encryption method for keys. Given current MD-5 speeds, this should permit transaction rates in the multiple hundreds per second. There are on the order of 3*10^8 seconds per year, so we should be able to support around 10^11 transactions per year. We'll divide by a factor of 10 to allow for idle time; this still leaves 10^10 transactions per year (contrastingly, a system involving a single public-key encryption per transaction using 768-bit keys will be lucky to reach 10^9 transactions). If each transaction at a broker is worth 10^-4 dollars (which is one percent of a penny transaction), the gross margin (I think that's the right term) is still 10^6 dollars per year, which should be enough to pay for the loaded cost of one computer. The transactions at vendors are much more profitable, although the load might be lower, and the transaction size is much smaller. Still, even at 10^-3 dollars per transaction, 10^9 transactions pays for the hardware and a person to keep it running; 10^10 transactions makes a lot of money for someone.

Extensions

By adding more information to Coin, vendors can use scrip as a mechanism to deliver coupons: you bought the first page of this article, so here's some scrip that buys you the second page at half price, if you use it in the next hour.

By using the techniques of scrip without brokers, even vendors that prefer a subscription model can prevent subscribers from allowing unrestricted use of accounts by others. Once someone gives away their scrip, they can't use it any more, because the serial number won't match. Thus, at the very least, this solves the problem of someone at a university posting their access code for subscription to a service for all to use, without requiring the imposition of severe rate limiting for subscribers.

As computers get faster, we can extend Millicent-like techniques into lower-priced service domains. At around the time that conventional techniques begin to be able to handle the problems of by-the-page information purchase, we'll be ready to handle the problems of paying for bandwidth by the hop, or paying for electronic mail forwarding. At least, we'll be able to handle these problems for source-routed networks.

Conclusions

The big new idea here is that cryptography can be useful to send yourself a message via an untrusted intermediary. Pushing the cost of storing infrequently-accessed information off to the client allows the servers to run quickly: the network acts as a perfect prefetch unit for loading your memory with the necessary state for the next operation. By way of example, a syndicated horoscope column may have tens of millions of readers; a server that had to store all of this information about subscribers might buckle under the seek time of accessing subscriber records. Pushing it off to clients allows us to have the salient parts of the database arrive in memory precisely when needed, at the small cost of verifying that the information hasn't been tampered with.

The secondary idea is that a service can help aggregate the transactions into a grain large enough to be acceptable to more conventional transaction handlers, without providing complete information about the transaction to the service. By making fund transfer explicit in each retrieval, all parties involved can instantly tell when they're being cheated; by limiting the scale of transaction, consumers can control the level of risk they accept.

Acknowledgments

In spirit, this work clearly owes great debts to lots of previous work in this field. Kerberos was certainly inspirational, as were efforts by DigiCash and First Virtual.

Smart Catalogs and Virtual Catalogs

Arthur M. Keller
Stanford University
Computer Science Dept.
Stanford, CA 94305 USA
ark@cs.stanford.edu

Abstract. We present an architecture for electronic catalogs, called Smart Catalogs and Virtual Catalogs. Smart catalogs are searchable, annotated combinations of machine-readable and machine-sensible product data. Virtual catalogs dynamically retrieve information from multiple smart catalogs and present this product data in a unified manner with its own look and feel, not that of the source smart catalogs. These virtual catalogs do not store product data from smart catalogs directly (except when caching for performance); instead virtual catalogs obtain current product data from smart catalogs to satisfy specific customer queries. Customers interact with smart catalogs and virtual catalogs through WWW or other interfaces. Product data is disseminated through the architecture using ACL (Agent Communication Language). In particular, ACL is used to communicate queries and answers among smart catalogs and virtual catalogs.

1. Introduction.

We present an architecture for smart catalogs and virtual catalogs that supports cross-search of multiple catalogs.

First let us consider how electronic catalogs are currently organized. Typically, these catalogs use ordinary World Wide Web (WWW) protocols and clients, such as Mosaic and Netscape. Each document, called a page, contains desired information or pointers to other pages, hopefully getting the user closer to the desired information. Typically, the World Wide Web (WWW) resembles a giant menu system, where each document behaves like a menu of links to other documents that hopefully get the user closer to the information desired. Each vendor maintains its own (collection of) catalogs. There may not be uniformity even among multiple divisions of the same company, let alone among multiple companies manufacturing or selling comparable products.

There are a variety of limitations of this typical approach to electronic catalogs. It is hard to find what you are looking for. You have to figure out where to start. For example, in order to find information about a particular product, you must separately visit each vendor of that product, and then navigate through that vendor's Web space to find the desired product. Making such navigation more difficult is that each vendor organizes its Web space differently.

Few catalogs have the ability to search by content (i.e., reverse-search). Reverse-search is typically by keywords. Keyword searching techniques, such as WAIS, are only of limited help, as they require that the user phrase the query using the terminology of the each data source, and vendors tend to use differing terminology from each other to describe products. A notable exception is Danish International's BingoSearch, which allows parameter-based search of a single catalog. Except for centralized catalog schemes that force a uniform catalog structure among multiple vendors, there is no fielded capability for cross-catalog search.

Our approach is to support reverse-search (i.e., search by content) of multiple vendor catalogs based on a deeper understanding of the contents of these catalogs.

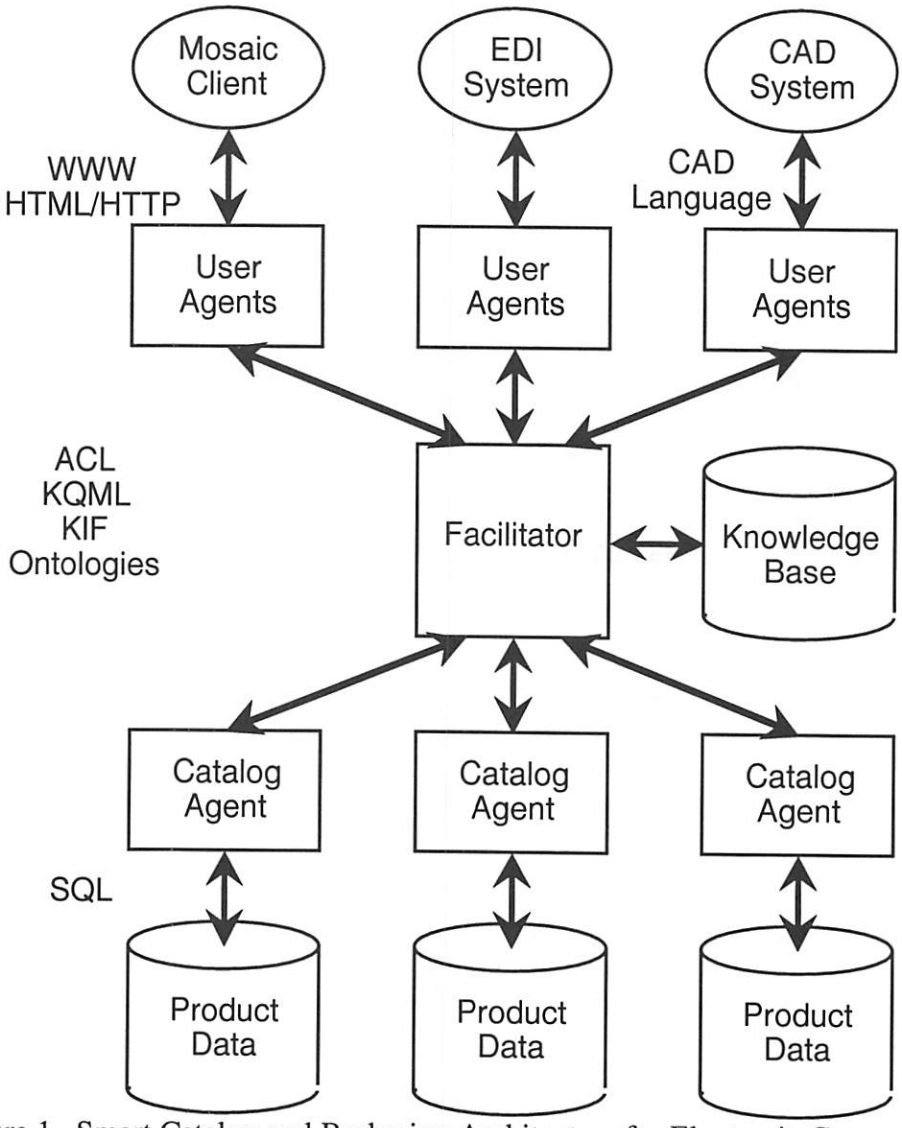


Figure 1. Smart Catalog and Brokering Architecture for Electronic Commerce

The following section describes the overall architecture of a Smart Catalog. Section 3 describes the concept of a Virtual Catalog. Section 4 contains a usage scenario of Smart Catalogs and Virtual Catalogs. Section 5 describes the current status and future work of this project. Section 6 gives our conclusions.

2. Architecture.

Our architecture is illustrated in Figure 1. It consists of agents communicating with facilitators, plus other components, such as data sources and user interfaces, that interact with the collection of agents and facilitators.

The communication language used by the agents in our architecture is Agent Communication Language (*ACL*). ACL consists of the Knowledge Query and Manipulation Language (KQML), the Knowledge Interchange Format (KIF), and a set of Ontologies. *KQML* consists of performatives, such as `ask-one`, `ask-all`, and `tell`, that describe the nature of the action to be taken. KQML has the role of the communication language in CORBA. *KIF* is based on First-Order Predicate Calculus and is the content language we use with KQML. KIF is powerful enough to contain or to encapsulate any other content language, so

that any information may be obtained from the information source, translated to the desired format, and transmitted to the requester, assuming the necessary components exist. Each product database is described by an *Ontology* (another term for ontology is "controlled vocabulary"), which defines the database, its structure, the terms used in it, and how they relate to each other. You can think of KIF as the "grammar," KQML as the "verbs," and ontologies as the "nouns" and "adjectives" that comprise the language for interoperating agents in our architecture.

Each Facilitator in our architecture acts as a broker. Facilitators perform routing, reasoning, and translation. A facilitator stores agent-provided advertisements of coverage in a knowledge base along with relevant ontologies. The facilitator uses these advertisements to determine which agents can support a particular request. And it translates requests into the language and terminology used by each responding agent and also translates responses into the language and terminology used by the requesting agent. A facilitator will decompose requests requiring action by multiple agents and then compose the responses for the requester.

Product data is stored in databases, for easy search and maintenance. Such data includes structured information, parameters, text, pictures, sound, video, etc. Each product database communicates with a Catalog Agent using its native language, such as SQL. The Catalog Agent performs 3 roles: It advertises the coverage of a product database; it understands queries and translates them into the language of the product database, and it packages answers from the product database in a standard format.

Someone using a WWW client, such as Mosaic or Netscape, will connect to a User Agent using the ordinary WWW protocols HTTP and HTML. The User Agent will present the user with an HTML form, either static or dynamically created. The user will describe the desired object using an HTML form. When responses come back from the facilitator, the User Agent will prepare dynamically created HTML documents with those responses.

We define four types of ontologies. *Base ontologies* are used to define common terms, such as engineering math, legal terminology, standard terms and conditions. Base ontologies are shared among all uses of this approach and are created by universities and research laboratories. *Domain ontologies* contain terms common to all or most vendors in an area, such as CPU speed, RAM size, or disk storage capacity. Typically domain ontologies are created by standards bodies and trade associations. *Product ontologies* contain company-specific terminology and refer to domain ontologies, such as NuBus cards for the Apple Macintosh. Individual companies create product ontologies, although other companies may refer to them. *Translation ontologies* are used to translate specific terms used in one ontology or information source to related terms used in another ontology or information source.* For example, a translation ontology may describe how to use an SVGA monitor (a PC standard) on a Macintosh. Individual companies create translation ontologies to enable them to compete in other markets. We expect there to be service organizations that create and maintain product ontologies and translation ontologies on behalf of other organizations.

3. Virtual Catalogs.

One important problem with using the WWW for product catalogs is the interaction between manufacturers and distributors or retailers. Consider a retailer that sells products from multiple manufacturers. The retailer will want

---

* The term "articulation axiom" is sometimes used to refer to entries in a translation ontology. Translation ontologies may be used to create an ontology algebra supporting translation across multiple ontologies.

to include product information from each manufacturer in its product catalog. Replicating all this product information in the retailers catalog would incur a considerable storage and maintenance cost. The natural approach using the WWW is for the retailer to hyperlink to each manufacturer's catalog so that the customer may obtain detailed product specifications.

There are several problems with the hyperlink approach. First, the customer may get "lost" within the manufacturer's webspace and not know how to get back to the retailer. Second, the manufacturer does not know the context of the customer's interactions with the retailer. Third, the customer may stumble upon a how-to-order page provided by the manufacturer, and wind up ordering from someone other than the original retailer. Fourth, if the customer does make it back to the original retailer by using the "back" button, no information determined at the manufacturer's site is carried along with the customer, such as the desired product configuration. Fifth, if the customer gets back to the retailer through the manufacturer's how-to-order page, the retailer does not know the original context of the interaction with the customer (e.g., other products selected for order in this same session).

One approach to multivendor catalogs is the integrated approach, where all the catalogs are stored on one site using one implementation. A notable example is by Open Market. An alternative is to provide some mechanism for manufacturers to respond to specific queries by retailers to satisfy customer requests for product information. This approach can be based on a business relationship between the retailer and the manufacturer. This approach is the one we take in a Virtual Catalog.

Virtual catalogs allow retrieval of product data using a distributor's catalog by combining information from multiple manufacturers' catalogs. This retrieval is performed dynamically, upon the user's request, based on the user's search criteria, using the terminology of the distributor or any connected vendor, and will retrieve data from any relevant connected vendor. Therefore, the distributor's virtual catalog is always kept up-to-date and in synchrony with each manufacturer's smart catalog. The distributor can choose to display all of a manufacturer's products or only a subset of those products.

With virtual catalogs, the distributor maintains control over the interaction with the user. The user never interacts directly with the manufacturer's catalog. Instead, the manufacturer's information is retrieved on demand and is presented to the user with the distributor's look and feel, not the individual look and feel of each manufacturer. The relationship between the user, virtual catalog, and smart catalogs is shown in Figure 2.

There are two key business relationships in the Virtual Catalog world, and many supporting business relationships. The two key business relationships are between the customer and the retailer (the virtual catalog), and between the retailer and the manufacturer (the smart catalog). Processes may exist in the virtual catalog to bridge these relationships. For example, orders from customers may trigger resupply EDI transactions to the manufacturer. Or the order may be forwarded to the manufacturer for drop shipment of the product directly to the customer. In addition, there are supporting business relationships. For example, if credit cards are used for payment, the customer has a relationship with the issuing bank, and the retailer has a relationship with the acquiring bank, and the two banks have a relationship for clearing the credit card charge. Similarly, there are relationships for order fulfillment, shipment, etc.

Virtual catalogs are appropriate for retailers and distributions, but they also enable new business models. A virtual distributor may operate using a virtual catalog and business relationships for order fulfillment, shipment, etc. The virtual distributor may not even have

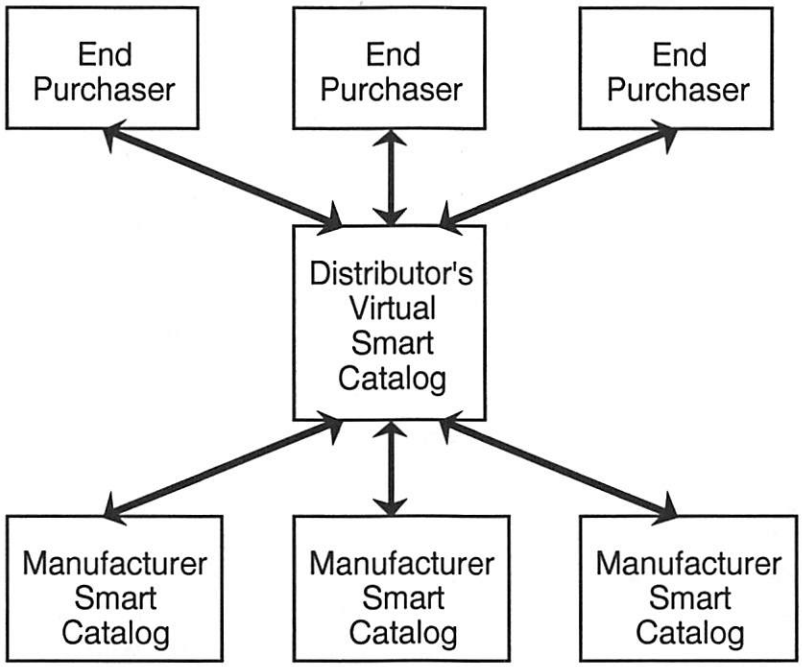


Figure 2. Virtual Catalogs.

any inventory, warehouse, etc. The virtual distributor could be a completely computerized setup, automatically providing product information using manufacturers' catalogs, taking orders and arranging for order fulfillment and payment.

4. Scenarios.

Consider a customer's request for color PostScript inkjet printers for the Macintosh costing under $1000. The User Agent will translate the query into KIF and submit it to a Facilitator. The Facilitator handling the query will consult its knowledge base for the facilitators or agents that can handle this request. For example, the Facilitator may transmit the request to the Catalog Agents for Apple and for Hewlett-Packard. The Catalog Agent will then interrogate the product database and translate the answer into KIF. For example, the Hewlett-Packard Catalog Agent may respond with the description of the HP 560C printer. The Apple Catalog Agent may respond with the Apple Color StyleWriter Pro. The facilitator will then collect these responses for the User Agent, which will package the responses in HTML for the Mosaic client. The user agent and facilitator are provided by the virtual catalog company. The catalog agents and product databases are provided by the manufacturers as part of their smart catalogs.

A customer may instead be interested in color PostScript laser printers for the Macintosh for under $3000. As of this publication, such printers cost around $5000, but prices are dropping. So the customer may request notification when any such printer is newly announced, or is lowered in price. This request will be stored in the relevant facilitator's knowledge base. When a manufacturer announces a new product, it will have a catalog agent send a message with the announcement and any changes to the advertisement of the agent to the facilitator. The facilitator will then send appropriate notifications to those parties who have expressed interest in this news.

Notice that the facilitator notification scheme is symmetric. Catalog agents express interest in being given product data queries. User agents express interest in being given product

data announcements. The same notification scheme is used for both types of activities.

Operators of virtual catalogs have several alternative models for paying for their operation. The operator may take and process orders and use the markup on the transaction. Alternatively, the operator may make money merely by providing information. (When information is free, search is valuable.) The operator may charge manufacturers, either a fixed fee or per referral. The operator may charge customers, either by subscription or per search. The operator may sell demographic information on customers or on searches to market research firms, manufacturers, retailers, or trade associations. Of course, an operator may obtain revenues from multiple of these approaches.

5. Current status.

The architecture of smart catalogs and virtual catalogs is an application of the facilitator architecture long in use in Logic Group of Stanford University's Computer Science Dept. The facilitator architecture has been demonstrated in the domains of software interoperability and concurrent engineering. It is now being applied to the domain of electronic commerce as part of Stanford's Center for Information Technology's (CIT) efforts on CommerceNet.

Several smart catalogs have been built in collaboration with several companies in the domains of workstations, test and measurement equipment, and semiconductors.

We are extending this work by creating a collection of smart catalogs for several manufacturers, a virtual catalog for a retailer, and a domain ontology for a trade association.

Please contact Arthur Keller by e-mail at `ark@cs.stanford.edu` or by WWW at `http://logic.stanford.edu/cnet.html` for more information.

6. Conclusion.

We have described an architecture for electronic catalogs for multiple companies that interoperate. Companies create smart catalogs of searchable, machine-sensible product information. Retailers and distributors create virtual catalogs that provide customers with product information dynamically requested from manufacturers' smart catalogs. Virtual catalogs provide a new degree of interaction between manufacturers and retailers or distributors. Virtual catalogs enable new business relations and new business models.

Acknowledgments.

This paper has benefited from feedback of numerous people who have heard presentations of this paper, read earlier drafts, or participated in discussions. In particular, Michael Genesereth designed the overall agent communication architecture used by smart catalogs and virtual catalogs, Narinder Singh designed the facilitator used here, and Mustafa Syed programmed some of the other components in our initial prototypes. Note an earlier version of this paper, co-authored by them, appeared at the Workshop on Electronic Commerce following CIKM, December 1994. Several other people have worked on the development of smart catalogs. These include Felix Chow, Bob Engelmore, Wanda Pratt, and Rupert Murdock.

This work was funded through a subcontract from the CommerceNet Consortium, which in turn derived its funding from a cooperative agreement with the U.S. Technology Reinvestment Program, as well as over 100 companies and organizations. In addition, several companies have provided funding of this project towards the construction of pilot smart catalogs.

References.

Danish International's BingoSearch can be seen at URL `http://www.danish.com`.

Michael R. Genesereth and Steven P. Ketchpel, "Software Agents," Comm. ACM, Vol. 37, No. 7, July 1994.

Thomas R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," in Nicola Guarino, Ed., International Workshop on Formal Ontology, Padova, Italy, 1992.

Open Market's URL is http://www.openmarket.com

William T. Wong and Arthur M. Keller, "Developing an Internet Presence with On-line Electronic Catalogs," Workshop on Electronic Commerce, December 1994, available from the URL `http://www-db.stanford.edu/pub/keller/1994/cnet-online-cat.ps`.

A Safe Tcl Toolkit For Electronic Meeting Places

(Extended Abstract)

Jacob Y. Levy and John K. Ousterhout
jyl@eng.sun.com, ouster@eng.sun.com
Sun Microsystems Laboratories
2550 Garcia Ave. M/S UMTV29-232
Mountain View, CA 94043

Abstract

Electronic commerce needs electronic meeting places to conduct business. To be useful, such meeting places must be safe for all participants and for hosts (owners of places). In this paper we discuss safety issues for participants and hosts. We then describe a system we are building, Safe Tcl, that will allow the construction of electronic meeting places with a range of safety properties. Safe Tcl has two attractive properties. First, it uses a simple security model based on ``padded cells'' that allows participants to coexist and interact safely. Second, Tcl makes it easy to integrate the numerous facilities required in an electronic meeting place such as integrity verification and authentication.

1 The Problem Of Safety

Electronic commerce, like human commerce, needs "safe places" where participants can meet to conduct business. The safety of a place can be measured by e.g.:

- Whether the host is protected against malicious or erroneous actions of individual participants.

- Whether participants are protected from each others' malicious or erroneous actions, and whether participants can be coerced by other participants to release, against their free will or without their knowledge and agreement, valued resources they carry with them (including information).

- Whether participants are protected from the actions of the host, both malicious and erroneous.

Tools for constructing safe meeting places for electronic commerce will become increasingly important as electronic commerce becomes more widely used. We believe that the basic security mechanisms for privacy, authentication, integrity checking and non-repudiation are relatively well understood. However, how to combine these mechanisms into higher level policies is less clear. Therefore, at this stage it is useful to create tools that allow experimentation and rapid prototyping as well as the construction and deployment of completed electronic commerce systems. Experience from human based commerce systems may be a useful guide in constructing electronic meeting and in choosing which tools to provide. We show how each safety problem identified above can be addressed in a computational context by drawing parallels from current common practice.

Currently, the human host and participants are protected from malicious intent of a participant by ensuring that no coercion tools (weapons etc.) are brought into the meeting place. Without a means for coercion there is no way for one participant to force another participant to release valued resources (such as the $1 million they are carrying in a briefcase) or information they own. Also, without means for coercion, there is no way for one participant to coerce the host to deny service or subvert its service to another participant. The equivalent in computational systems is to place each participant (or group of mutually trusting participants) in a separate environment (``padded cell''), thus restricting their ability to manipulate the state of other participants or the host. Functionality in an environment is restricted to remove any method for a participant inside the environment to harm another participant outside the environment. To enable communication between participants, environments are extended with controlled communication channels that only allow legitimate communication.

Protecting a human participant from the host is currently achieved through insurance and liability based mechanisms. Upon entry into a meeting place the participant is at risk of being coerced by the owner of the place to divulge information or to part with valued resources. These risks can be ameliorated by insurance or liability shifting arrangements, or by bonding. Similar mechanisms can be implemented in an electronic commerce system: a third party can offer insurance covering aspects of electronic business such as compromise of a transaction or participant owned resources by a host. Since these mechanisms are based on authentication, integrity checking and privacy, a system that provides access to these building blocks suffices.

2 Tcl And Tk

Tcl [1] is an interpreted scripting language originally developed by Ousterhout at UC Berkeley, widely used for rapid application development. Tcl is easy to extend and one of its most popular extensions, Tk, is very pop-

ular for rapidly constructing highly interactive GUI components for programs.

Tcl programs are portable between systems on which a Tcl interpreter is implemented. Tcl programs are strings, as is all data manipulated by a Tcl program. Tcl programs can thus be input data for other Tcl programs, and Tcl programs can be generated by Tcl programs and executed on the fly or transported in textual form. The global data state of a Tcl execution can be obtained and saved as a Tcl script, which when executed, restores the state to what it was when the script was created.

3 Borenstein's And Rose's Safe Tcl

Safe Tcl is an extension of Tcl created by Borenstein and Rose [2] to allow safe execution of programs transported by email. It takes advantage of the ability of Tcl to support multiple totally independent Tcl interpreters within a single application process. Each Tcl interpreter provides a separate name space for variables and procedure definitions and a separate call stack for procedures. The command set of each interpreter can be tuned to include exactly the commands desired.

Borenstein and Rose's Safe Tcl provides two interpreters: one is unrestricted, and the other's command set has been restricted to exclude unsafe facilities such as access to files and creating subprocesses. Incoming scripts can be executed in the restricted interpreter without fear of damage to the environment. The application's unrestricted intepreter can implement safe methods for access to unsafe facilities and export these to the restricted interpreter by ``declaring them harmless''. The overall effect is much like the common separation in operating systems between kernel space and user space: the unrestricted interpreter corresponds to kernel space, the restricted interpreter corresponds to user space, and the features that are declared harmless parallel system calls. Safe Tcl is more flexible than the traditional kernel-user space mechanism because features exported to the restricted interpreter can be modified at run-time.

4 The New Safe Tcl

The Safe Tcl extension created by Borenstein and Rose is useful for executing simple untrusted scripts. However, because it allows only one untrusted interpreter to exist at a time it is not powerful enough to host mutually suspicious scripts and it does not provide communication mechanisms between such scripts. Also, it is intimately tied to email as a transport mechanism, and it does not provide security features such as authentication or integrity checks.

At SunLabs we have generalized Borenstein and Rose's ideas to allow for multiple restricted interpreters, called ``slaves'', under the control of a ``master'' interpreter. A slave can be master for other slaves, forming a hierarchy of interpreters. A master interpreter can configure the command set of its slaves to be a proper subset of the features that it itself has access to. A master can execute arbitrary scripts in the slaves. A slave has no access to its master except as described below.

We have also generalized the ``declare harmless'' mechanism of Borenstein and Rose's Safe Tcl into a more general ``alias'' mechanism for communicating between interpreters. An alias creates a link between two interpreters such that when a particular command is invoked in one, the source interpreter, another command is executed in the other, the target interpreter. The target command receives the arguments of the source command, and the result of the target is returned as the result of the source command. A master interpreter can create aliases between its slaves and itself, which are analogous to the ``declare harmless'' mechanism of Borenstein and Rose. A master can also create aliases between any of its slaves in order to allow them to communicate directly.

Our third generalization will be to incorporate authentication techniques [3, 4] into Safe Tcl. A master interpreter will be able to authenticate scripts before executing them in slave interpreters. Based on the trust assigned to the script's author, the master can customize the facilities available to the slave. A master intepreter will also be able to authenticate aliases individually, so that a script can include both untrusted parts that must execute in a slave interpreter and trusted parts that form aliases to be executed in the master.

Our final generalization is to separate how to execute untrusted scripts from how they are transported. We feel that transport issues are not a part of the problem to be solved by Safe Tcl and that we should not innovate here. We will connect Safe Tcl to email, HTTP and other mechanisms to allow participants to enter and leave places through a variety of mechanisms. Also, how participants find and name each other is not a problem solved by Safe Tcl. Safe Tcl will be able to use a wide choice of external naming schemes. Of course, for Safe Tcl to be widely useful, eventually a small set of core transports and naming schemes must be supported. However, at this time we are building for experimentation and thus we will provide no built in naming or communication schemes.

Safe Tcl offers two advantages in setting up electronic meeting places. First, its security model is very simple and makes it easy to isolate mutually suspicious participants. Safe Tcl is based on the kernel-user model that has been used successfully for thirty years in time-sharing systems, and it provides considerably more flexibility. Safe Tcl is implemented as part of the Tcl interpreter, so it offloads many security concerns from the operating system kernel to the Tcl interpreter. For

example, Safe Tcl ensures that Tcl programs can not modify other programs' address spaces even on insecure systems such as some personal computers' operating systems which do not provide this assurance.

The second advantage of Safe Tcl stems from the Tcl scripting language, which makes it easy to integrate a variety of components. Electronic meeting places for commerce must include a large number of mechanisms from electronic cash to encryption to databases. With Tcl it is possible to extend the system facilities with code written in C or to write Tcl scripts to glue existing components together. Tk allows electronic participants to interact with humans as well as with other participants. Since Tk interfaces are Tcl scripts, they can be part of a participant's script.

5 Electronic Meeting Places in Safe Tcl

Separate interpreters allow mutually suspicious and antagonistic participants to be in one meeting place while preventing them from harming each other or obtaining access to each other's private data. Because they are executing in separate interpreters from that of the host, participants can not harm the host. The host controls the creation of communication channels that allow participants to communicate with each other and with the host. These channels can be audited to provide accountability and non-repudiation. The host can also grant controlled access to a participant to specific safe uses of unsafe functionality on an as-needed basis. Using resource allocation policies implemented with tools provided by Safe Tcl, a host will be able to limit resource usage by scripts executing in its slave interpreters to prevent denial of service attacks or resource hogging.

The roles of host and participant are fluid. Using the tools provided by Safe Tcl a participant can construct a temporary meeting place for its own use while it is visiting a place that allows this. It can control which participants are admitted to its place and all aspects of their computation while the participants are within its own place.

Protecting participants from a host is a difficult problem. It is fundamentally impossible to protect a program from malicious behavior on the part of the engine that executes it. The host controls the engine (the Tcl interpreter) that executes the particpants, so there is no way to protect participants from a malicious host. A viable approach is for participants to ``avoid bad neighborhoods'' by only entering meeting places that are known to be trustworthy. For example, trustworthy meeting places could be independently certified and identify themselves with well-known public keys. A would-be participant can issue a challenge to the host with an advertized public key and only enter the place when the challenge is satisfied. Also, participants can insure themselves or require hosts to be bonded. Safe Tcl can be integrated with the needed mechanisms to enable this scenario.

6 Remaining Open Problems

Here are some remaining open problems that Safe Tcl currently does not address:

How to implement electronic currencies. There are several well known approaches to solving this problem [5,6]; we believe that interfacing Safe Tcl to one or more of these is simple, and we will do so in a future version of Safe Tcl.

Persistence and recoverability of participants. Safe Tcl does not afford participants protection against crashes of a host; all state and information carried by a participant will be lost if a host crashes or disallows the computation of a participant to complete. This is a hard problem, a sub-problem of protecting a participant from a place's host, and will be solved in that context.

Resource Controls and Denial of Service Attacks. Safe Tcl itself does not provide any mechanism for accounting for resources used by an application, and for preventing an application from mounting a denial of service attack through excessive resource consumption. However, the clean separation between applications through the use of separate interpreters affords a clean starting point for building a resource accounting system. A future version of Safe Tcl will explore these issues.

7 Availability

Safe Tcl is available by anonymous FTP. Retrieve the file ftp.smli.com:/pub/tcl/stcl0.2.tar.gz. The software is also available at the Tcl archive managed by Alcatel Inc., at ftp.aud.alcatel.com:/tcl/extensions/stcl0.2.tar.gz.

8 References

[1] J. K. Ousterhout, "Tcl And The Tk Toolkit", Addison-Wesley, (1994).

[2] N.S. Borenstein and M. Rose, "Safe Tcl", available at ftp://ftp.fv.com/pub/code/other/safe-tcl.tar.Z.

[3] S.Garfinkel, "Pretty Good Privacy", O'Reilly & Associates, Inc., (1995).

[4] W. Diffie and M. Hellman, "Privacy and Authentication", IEEE Proceedings, pp. 397-427, (1979).

[5] D.Tygar, "NetBill: An Internet Commerce System Optimized for Network Delivered Services", IEEE CompCon '95, (1995).

[6] D.Chaum, "Online Cash Checks", Advances in Cryptology EUROCRYPT '89, pp. 288-293 (1989).

Developing and Deploying Corporate Cryptographic Systems

Diane E. Coe (dec@mitre.org)
Judith A. Furlong (jfurlong@mitre.org)

The MITRE Corporation
202 Burlington Road
Bedford, Massachusetts 01730-1420

The MITRE Corporation has developed a number of cryptographic systems that provide security-related capabilities for its growing electronic information environment. These cryptographic systems were developed using commercial off-the-shelf products and widely accepted standards. Standards were adhered to in order to assure interoperability with emerging products such as those used to conduct electronic commerce on the Internet. However, there are many shortcomings within the marketplace and the Internet community that are limiting the extent to which interoperability can be achieved. Products are not as "open" as we need them to be, infrastructures are not in place, and standards are still evolving. This paper presents many of the issues with which we are faced and, in some cases, provides solutions. MITRE's needs are not unique. They are the needs of any enterprise wishing to provide an integrated security solution for the access and exchange of valuable electronic information.

This paper first introduces The MITRE Corporation and describes its corporate cryptographic systems and information infrastructure. The paper then discusses issues associated with integrating MITRE's cryptographic systems with emerging electronic commerce products both for use within the corporation and for conducting electronic commerce externally. The paper concludes with recommendations and lends some insight into where The MITRE Corporation is headed in terms of its corporate cryptographic systems and electronic commerce.

The MITRE Corporation

The MITRE Corporation is a multifaceted engineering organization providing technical and strategic guidance in communications, information, and environmental systems. MITRE's expertise encompasses a broad range of disciplines including computer science, electrical engineering, mathematics, management, and administration. Serving military, government and civilian needs worldwide, MITRE supports a diverse set of national and international clients.

In order to provide for more efficient operations, MITRE is transitioning many of its paper-based business applications to an electronic environment. With this transition comes changes in business practices. Practices which were applied in a paper-based environment may no longer be applicable or implementable in an electronic environment. Existing business practices must be examined to determine which practices are still needed in an electronic environment and to identify which practices need to be modified to work in an electronic environment.

One business practice that is needed in both a paper and an electronic environment is an approval capability. In the paper world, approval is usually given through a handwritten signature applied to a document, form, etc. An analogous approval capability is needed in the electronic world. Such an approval capability may be implemented using a cryptographic mechanism called digital signatures.

Digital signatures provide not only an approval capability but also the security services of authentication, integrity, and nonrepudiation. Thus, digital signatures offer more capabilities than its handwritten counterpart, which cannot ensure the integrity of information after it is signed.

A second change that is occurring at MITRE is in the architecture of its information systems. In the past, a few mainframes were used to support all of MITRE's corporate applications. However, new applications are client-server based and, while the mainframes are being phased out, the number of servers in use by the corporation has increased dramatically. Users of these systems, who had a small number of account names and passwords to keep track of in the past, now have more than they can possibly remember. In addition, with more and more valuable information and assets being managed and stored electronically, the need to protect authentication information has taken utmost priority. These changes have caused a revisitation of the authentication mechanisms in use at MITRE.

The following sections describe the digital signature capability that has been developed at MITRE, the key management system that will support the digital signature capability, and the authentication system that is being deployed throughout the corporation.

MITRE's Digital Signature Environment

Digital signatures are being implemented at MITRE to decrease both the amount of time and the amount of money spent on processing and archiving paper forms. The first corporate application that will provide a digital signature capability is an application known as Data Capture. The Data Capture initiative comprises a set of electronic forms called transactions, a digital signature capability, and a work flow system. With Data Capture, forms are generated, signed, and routed for approval electronically. Information is captured in a central database when it is first created. Rekeying of data is eliminated and data is reusable, thus eliminating the need to type the same information over and over. Data Capture significantly reduces typographical errors, both by virtue of the fact that rekeying is eliminated and because validity checks are automatically performed on the electronic data.

Data Capture is a client-server application. It is written in C++ and accesses a relational (SQL-based) database. Data Capture clients run on both Macintosh and Windows 3.1 platforms and will run on additional platforms in the future. The server and database reside on a UNIX platform.

Using a commercially available toolkit from RSA Data Security Incorporated called BSAFE, MITRE developed two digital signature application programming interfaces (APIs) that are called by the Data Capture application. One API performs a signing function, and the other performs a signature verification function. These APIs are written in ANSI C and easily port from platform to platform. They can be called by Data Capture or by any forms package or development environment that provides a C interface.

Before an employee can digitally sign a Data Capture transaction, the employee must obtain a public-private key pair. With RSA public key cryptography, the keys in the pair are mathematically related in such a way that what is encrypted by one key can be decrypted by the other and vice versa. Yet, the private key cannot be easily derived from knowledge of the public key. Together, these two attributes are what allow the technology to be used for digitally signing information. When the user obtains his key pair, he keeps his private key private and uses it to encrypt information, thus applying his "signature" to it. He makes the public portion of the key pair publicly available so that anyone can retrieve it and decrypt the information he has encrypted. If the recipient of the signed information can successfully decrypt the information using the signer's public key, he is assured that the information was, in fact, signed (i.e., encrypted) with the signer's private key. If the signer has not divulged his private key to anyone, he is the only one who could have produced that signature. The key management system that supports MITRE's digital signature capability is described in the following section.

Once a user has a key pair, he is ready to sign. With the Data Capture application, all data appearing on a form is signed by the employee. This does not have to be the case and, in the future, will not be. In subsequent versions of Data Capture, only certain data fields appearing on the form will be signed by employees acting in particular roles. As an example, on a Purchase Requisition, an employee's manager may sign only fields that describe the items being purchased, thus signifying approval of the items, while someone in the budget office may sign the Project Number and Total Cost fields, thus signifying that the identified project can support a purchase of that amount.

Because encryption using public key technology is relatively slow, the data to be signed is first sent through a hashing function. This function reduces the data to a 128 bit "fingerprint" of the information, regardless of the amount of data to be signed. The hashing algorithm is strong enough to detect even a single bit change of the data—that is, if any bit of the data is changed, a different hash value will result. It is this 128 bit hash value that actually gets signed, or encrypted, using the employee's private key. The encrypted hash is the employee's signature on the data. The hashing of the data provides integrity (i.e., any change of the data will be detected) while the signature provides authenticity and nonrepudiation. The signature is stored in the database with the data with which it is associated.

MITRE's Key Management System (MKMS)

In order to support digital signatures, a key management system is needed. MITRE's key management system (MKMS) plays a pivotal role in the generation, distribution, and management of the cryptographic keys that each MITRE employee needs to utilize the digital signature capability. The MKMS also provides protection of these cryptographic keys. The confidentiality of the private keys is provided through encryption. Digital signatures on public key certificates, which bind users' identities with their public keys, assure that the integrity of the public key contained within the certificate is maintained.

At MITRE, employees obtain their key pairs from centrally-located Key Generator systems. There are multiple Key Generator systems to service all of MITRE's sites. The Key Generator systems are operated by Corporate Security representatives. Each employee presents himself to the representative who authenticates the employee and directs him or her to run a simple key generator program. This program first prompts the employee for his or her employee id. The employee id is used to obtain the employee's distinguished name from MITRE's X.500 directory. The distinguished name is later used in generating a certification request. Once the name is obtained and the employee confirms it is correct, the employee is then prompted for a password. The employee-chosen password is used to encrypt the employee's private key before it is stored on a floppy disk. The floppy disk is readable by both the Windows and Macintosh versions of Data Capture. (An effort is underway to transition from a floppy disk medium to a multipurpose smart card. This effort is discussed in the following section.)

The key generator application then creates a certification request adhering to Public Key Cryptography Standard (PKCS) #10, *Certification Request Syntax Standard*. The certification request is signed by the Corporate Security representative to attest to the employee's identity. The signed certification request is sent to MITRE's Certification Authority (CA) for processing via a server program. The key generator program does not maintain a copy of employees' passwords nor private keys.

There are several reasons why a centralized key generation scheme was implemented. While the key generator application could be made available to individual employees, each employee would still need to present themselves to Corporate Security before their public key certificate is generated. (Corporate Security is responsible for attesting to the employee's identity and ensuring that the certificate request contains that employee's distinguished name.) In addition, the logistics involved in maintaining a few instantiations of the key generator program and its external interfaces (e.g., an X.500 interface is required to retrieve employees' distinguished names) seemed much simpler than maintaining a corporate-wide disbursement of the program. In time, the software may be rearchitected and given to end users.

As stated previously, private keys are on floppy disks. Employees are advised to protect the disk as they would other important personal information and are discouraged from recording their password or storing their keys on unprotected Macintosh or PC machines. The disks are a distinctive color so they are easily distinguishable from other disks. Floppy disks are considered an interim solution that provide a low-cost, platform-independent capability. Employees can take their disks with them and can sign from their home computers or while traveling. Because the signatures being applied in the initial versions of Data Capture are not critical, in that they cannot obligate significant sums of money or provide access to highly sensitive data, this interim capability is considered acceptable. The transition from a floppy medium to a smartcard will provide enhanced protection for the private keys in the future.

Public keys are sent to MITRE's CA in PKCS #10 format to be put into signed certificates. Certificates bind an employee's identity with his public key. The certificate contains identifying information about the employee and his or her public key. The binding takes place when a trusted entity (i.e., the CA) signs the certificate. Just like any other information that has been signed, the signature on the certificate can be verified using the signer's (i.e., the CA's) public key. Every employee has a copy of the CA's public key. Use of signed certificates guards against masquerading.

Once a public key certificate is generated, there must be a mechanism for revoking the certificate should the affiliated private key be lost or

compromised. For the initial implementation at MITRE, the CA operator is notified by a corporate security representative when a certificate needs to be revoked. The CA operator uses a function within the CA system to indicate that the certificate has been revoked.

A question that needed to be resolved was how will the certificates and the CRLs be made available to individuals and applications that need to access them? Initially, it was envisioned that all certificates and CRLs would be stored in the corporate X.500 directory server. X.500 provides an easily-accessible storage location for employee information. There are, however, drawbacks to using an X.500 directory in a corporate environment.

At MITRE, digital signatures applied to corporate information may need to be verified relatively soon after they have been applied or they may need to be verified days, months, or even years later. In the X.500 directory, certificates are stored in a "last in, first out" order within a user's X.500 entry and are not individually addressable. If a user has multiple certificates (e.g., a current certificate and some number of expired and/or revoked certificates), a query against that user's entry in the X.500 directory will return all of the user's certificates. The application would then need to parse through each of the certificates to find the certificate that was valid at the time the signature was applied. Once the appropriate certificate is found, a check needs to be made to ensure that the certificate was not revoked prior to the time the signature was applied. To do this, the application must first extract the serial number from the retrieved certificate. It then performs another query against the X.500 directory. This second query is against the certification authority's (CA's) entry in the directory and retrieves the certificate revocation list (CRL). If the CA's entry contains multiple revocation lists, the application must parse through the retrieved CRLs and find the first list that was produced after the signature in question was applied. The application then searches through the CRL to determine if it contains the serial number of the retrieved user's certificate. If it does, the date of revocation is checked; and if the revocation date is prior to the signature date, the signature is considered invalid. If the revocation date is later than the signature date or if the certificate is not on the CRL, the certificate is valid. This is quite a bit of processing to be performed every time a signature needs to be verified.

To reduce the amount of processing that has to be performed to verify a signature, a decision was made to store certificates in MITRE's data warehouse which is managed by a relational database management system (DBMS). The tables containing the certificate information can be read by corporate applications but updates to the tables are controlled by a secured maintenance account. When a certificate is created by the CA system, it is sent to a server program known as the KG/CA server. This server stores the certificate in the data warehouse, indexed by the employee's unique employee number. The validity dates of the certificate are stored as individual fields in the table, as well, so that queries can be performed against them. CRLs are also sent to the KG/CA server. The server parses the CRL and updates the records in the certificate table to indicate the date a certificate has been revoked. This simplifies the signature verification process significantly in that now a single query can be used to determine whether or not a valid certificate exists. Certificates and CRLs will continue to be stored in the X.500 directory, but only the user's current certificate and the current CRL will be stored. It is envisioned that the X.500 directory will be used in the future for sending and verifying secure messages and for communicating securely with external entities.

Distributed Authentication

MITRE's rearchitecting of its information systems has caused an increase in the number of corporate servers and, thus, the number of accounts each user has. Passwords are currently the common method of user authentication to MITRE's servers. Typically, a user must supply a login identifier and a password to gain authorized access to a particular computer system or host. In most cases, passwords are user selectable, although restrictions apply. User-selected passwords are often easy to remember which renders them easy to guess. In practice, assigned passwords are usually difficult to remember and are often written down, making these passwords vulnerable to unauthorized disclosure. To avoid having to remember multiple passwords for different systems, users often use the same password on every system to which they have authorized access. Thus, multiple systems are at risk under a single or common attack. The high

volume of abuse regarding the selection of user passwords combined with the failure to administer user password protection successfully has resulted in the high demand for unitary authentication in MITRE's distributed computing environment.

MITRE has an effort underway to deploy a distributed authentication system based on Version 5 of Kerberos. Kerberos is a trusted third-party unified network authentication service. It is designed to operate in a client/server environment. In a "kerberized" environment, users are known as principals. A list of authorized users, or principals, is maintained in the Kerberos principal database. Once a user is known to the principal database, he or she can get what's known as a Ticket Granting Ticket (TGT) from the Kerberos Key Distribution Center (KDC). A TGT is a credential that will allow the user to obtain other credentials, or tickets, for kerberized services within the user's Kerberos environment. The user's environment is called his realm. A realm consists of all the users, services, and hosts listed in the Kerberos principal database.

To get a TGT, the user provides his user or principal name to the KDC. The KDC then sends the user a TGT, but portions of the TGT are encrypted in a password known only to the user and the KDC. The user enters his password and decrypts the TGT rendering it usable. This is the only time that the user must provide authentication information. All subsequent authentication within the Kerberos environment is performed by Kerberos on behalf of the user. This provides a unitary authentication capability. Note too, that because decryption of the TGT is performed at the user's workstation, his or her password never traverses the network.

Once a user has a TGT, he can present the TGT to the KDC and receive tickets for kerberized services. The product MITRE is investigating for widespread deployment provides a number of kerberized services, including a kerberized *rlogin*, which allows users to remotely log in to UNIX hosts; *rsh*, which allows users to execute a command on a UNIX host; and *rcp*, which allows users to copy files to and from UNIX hosts. In a nonkerberized environment, each time a user wishes to access one of these services, he has to authenticate himself to the service or host. With Kerberos, tickets are used for authentication and are passed on behalf of the user to the service or host. No other authentication is required.

Some of the biggest proponents of distributed authentication are system and database administrators (DBAs). At MITRE, DBAs administer databases residing on as many as six different machines. In order to perform administration and testing, the DBAs often log into the systems in different roles, thereby obtaining different privilege sets. Given that there are approximately four roles that a DBA may assume on each of the six machines hosting database management systems, there is a potential twenty-four account names and passwords that a DBA must remember. Kerberos reduces this number to one.

MITRE is currently evaluating Kerberos for potential deployment throughout the corporation with DBAs being the first set of users. The prototype environment currently consists of Macintosh, PC, and UNIX Kerberos clients and a KDC and principal database residing on a SunOS machine. Kerberized services are resident on SunOS, Solaris, and AIX machines and are currently being installed on a Sequent machine. This prototype environment mirrors the corporate environment and will be used as a testbed until the production system is deployed.

Integrated System

As described in the previous sections, MITRE is actively pursuing the integration of security services into MITRE applications. Capabilities presently being examined are digital signatures and distributed authentication. Future capabilities may include secure mail and data encryption. The implementations of these capabilities require that MITRE users possess certain credentials or information. In the case of authentication, MITRE users need authentication credentials such as Kerberos tickets; while in the case of digital signatures, users need public/private key pairs. Each of these security capabilities are being independently developed, and each require some amount of support information. The goal of the MITRE single token task is to consolidate the support information for all the security services on a single token. In some instances, the actual security service may be performed on the token. In other instances, the token will provide secure storage for the information. MITRE's single token will be an ISO smartcard.

A number of capabilities will ultimately be supported by the single token. The token will support MITRE's digital signature capability by providing a more secure storage media for private keys. Data will be sent to the card to be signed so the private key will never be vulnerable in the workstation. MITRE's digital signature APIs have already been modified to access the ISO smartcard, and a prototype smartcard-based signature capability has been developed.

The Kerberos-based distributed authentication capability is being incrementally integrated into the MITRE environment. The integration of the Kerberos authentication protocol at MITRE is the first step towards achieving a single sign-on capability. MITRE is working with its Kerberos vendor to develop a smartcard interface for the Kerberos product.

Kerberos is not a complete solution for achieving single sign-on. Kerberos currently provides single sign-on for only some UNIX servers. A full single sign-on capability cannot be achieved until a means for handling the passwords associated with nonUNIX systems is implemented. Toolkits for developing scripts are being investigated as a means for storing passwords on the smartcard, thus moving us closer to our goal of a truly single sign-on capability. The token can support the single sign-on capability by storing various types of authentication credentials so the user will not be required to memorize and supply authentication information each time he wishes to connect to a MITRE system.

In a fully-automated corporate environment, restricted information must be electronically accessible by authorized users. In order to support this capability, MITRE must add encryption to its environment. With encryption, restricted information such as salary information or personnel information can be kept confidential while it is transferred to computers throughout the corporation. The token can support an encryption capability by providing secure storage for the keys used in the encryption process. It is also possible that the token could perform some of the necessary encryption functions.

MITRE presently uses a challenge-response mechanism to provide access across its firewall. It is hoped that the single token system can also replace this mechanism.

Another important use of the token will be for facility access. Currently, a badge is needed to allow MITRE employees access to MITRE facilities and restricted areas within such facilities. To reduce the number of tokens required by MITRE employees to conduct their daily business to one, the MITRE badges will be replaced by the ISO smartcard. Replacement of badges will also provide some assurance that employees won't leave their smartcards in unattended systems since they must have their badge on their person at all times.

MITRE Information Infrastructure

While the aforementioned systems are being developed by MITRE's Information Security Division, the Information Systems Division is developing MITRE's Information Infrastructure (MII). The MII is an electronic information system based on World Wide Web technology and will be the primary source of information at MITRE. At the present time, no restricted information is posted on the MII. All information residing on the internal Web servers is accessible to all MITRE employees, and all information residing on MITRE's external server is publicly releasable. In order to post information having dissemination restrictions on these systems, security in the form of access control and authentication is required. This security is necessary to protect the corporation's valuable information assets from unauthorized disclosure and to protect the privacy of employee-related data.

Integration of the MII with Corporate Cryptographic Systems

MITRE began investigating secure Web products in February of this year to determine whether or not they could provide the security MITRE needs. The ideal product would meet the MII's security requirements by using one of the cryptographic systems already in place or in development. Products were assessed to determine the extent to which the products could be integrated with MITRE's public key infrastructure or its Kerberos environment. Most of the products found were public key-based and implemented either the Secure HyperText Transfer Protocol (S-HTTP) or the Secure Socket Layer (SSL) protocol. One Kerberos-based product was also found. Given the number of security-enhanced products that existed, our assessment began on an optimistic note.

The optimism quickly faded. It turned out that although S-HTTP servers existed, the protocol was not supported in any of the commercially-available browsers. The SSL protocol had both browser and server implementations, but the products could not be integrated with our existing public key infrastructure. The public and private key pair of the SSL server must be generated by software embedded within the server. Key pairs generated by MITRE's Key Generator system could not be used, nor could a certificate signed by MITRE's certification authority because the products only recognize certificates issued by a small set of CAs and the certificates associated with those CAs are embedded in the client software. This means that although MITRE is purchasing thousands of certificates to support its digital signature capability, it could not use any of these certificates for its Web servers.

There were other limitations found in the investigated products that hampered integration with MITRE's public key infrastructure. Systems supporting public key technology had the private key of the server encrypted in software and stored on the server. There was no flexibility to store the private key on an external storage device such as an ISO smartcard or even a floppy disk as is done at MITRE.

Client authentication and access control in the Web products was also a problem in terms of providing a capability consistent with MITRE's goals. Because there were no products supporting public key-based authentication for clients (only for servers), authentication is performed using passwords. MITRE is trying hard to eliminate the number of passwords users have to remember. Implementation of password-based authentication in our Web server is contrary to our single sign-on goal.

The Kerberos product proved promising from an integration perspective, but it also had its shortcomings. On an optimistic note, we were delighted to find that the Web application, which implements Version 4 of Kerberos, could, in fact, obtain tickets from our Version 5 Kerberos server. However, the client we tested stored the retrieved tickets in a nonconfigurable location, thus we were not able to use the same credential cache for both applications. This necessitates logging on to our Kerberos server twice; once to get a Ticket-Granting-Ticket (TGT) that can be used to obtain a ticket for our Web server, and once to get a TGT to get tickets for all our other applications. So the integration of the kerberized Web product with our Kerberos environment wasn't fully successful, but it came much closer than the public key-based products.

External Use of Corporate Cryptographic Systems

It has always been a goal within MITRE to develop cryptographic systems that are expandable to allow for secure external communications, as well as to provide security within the corporation. Our cryptographic systems are based on standard technologies and protocols. We have public key certificates that follow the X.509 standard; we are storing our certificates in an X.500 directory server; our digital signatures are created using the MD5 hashing function and the RSA algorithm; our distributed authentication system is a standard V5 Kerberos implementation; and our token is an ISO smartcard. We have followed all the guidelines, implemented all the standards, and developed open systems, all in an effort to ensure interoperability with the rest of the world; and with whom have we achieved interoperability with respect to our cryptographic systems? That remains to be seen.

Probably the biggest shortfall in the community today, the one area most hampering the widespread use of public key technology, is the lack of a ubiquitous public key infrastructure. Public key cryptography is less than two decades old. Infrastructures to manage public keys and certificates in large environments have only recently been developed and are just now being implemented. Not all the issues associated with public key and certificate management have been resolved. Competing approaches to public key management along with the need to support different public key algorithms have resulted in several different public key infrastructures being in existence today. The Internet (PEM) Key Infrastructure, the NIST Public Key Infrastructure, and the MISSI Infrastructure are three of these infrastructures. It is unclear whether a single public key infrastructure that services the needs of all communities will exist in the future. Because of this, organizations are developing their own custom hierarchies, and trust among hierarchies does not exist. Certificates generated within one environment cannot be authenticated by a recipient in another environment because the environments

do not have a mutually trusted CA. Certificates generated by MITRE are not going to be accepted by Visa or Mastercard or by any of the many companies now conducting electronic commerce on the Internet. Each will either issue a certificate signed by its own CA or develop its own method for providing authentication and nonrepudiation services. One method currently being used is a call-back mechanism. Users placing orders are sent confirmation requests to their registered electronic address. Orders are not filled until the confirmation is received. This prevents anyone except those having access to the user's electronic mailbox from masquerading as the registered user. Ad hoc trading is not being supported. The only trading that is occurring is between parties having preestablished relationships.

Even in the world of EDI, bilateral trading agreements are negotiated out of band and in advance of any EDI orders being accepted. Currently, there is little security within EDI transactions. Rather than securing the transaction, threats are being dealt with by mutually-accepted policies established in trading partner agreements or by the systems processing the transactions. The threat of receiving orders from someone masquerading as an authorized trading partner is being mitigated by the implementation of rules at the receiving system that detect and flag anomalous orders. We have yet to find a company among the fifty thousand currently conducting business using EDI that uses digital signature in its EDI transactions. Products exist to support digital signatures in EDI, but is anyone using them? It does not seem possible with the ubiquitous hierarchy.

Systems supporting Kerberos are also emerging in the world of electronic commerce. Can we authenticate ourselves to these systems using our Kerberos credentials? No. Kerberos credentials are only recognized within the realm in which they were issued. Inter-realm authentication does not quite yet exist in the community.

Conclusion

In spite of it all, we are still pretty excited about the cryptographic efforts ongoing at The MITRE Corporation. We have learned a lot of valuable lessons and have gained insight and experience in many different areas. We have made great strides in working towards our goal of a paperless environment and are providing enhanced security in our electronic systems as well. In the end, we believe we will have a more efficient and more secure workplace and lower operational costs. However, we have some recommendations that will help, not only us, but also companies and agencies, both government and civilian, with similar goals and desires of which we believe there are many.

Products need to be more open. They should allow for the use of keys generated external to the product's environments. Flexibility should be provided to allow private keys, in particular, to be stored on an external storage device such as an ISO smartcard or even a floppy disk.

Certification authorities, key lengths, and algorithms should not be hardcoded into the products. RSA is not the only public key algorithm in existence today which may be used for digital signature and key exchange. Other digital signature and key exchange algorithms include the Digital Signature Algorithm (DSA) specified in FIPS 186 - Digital Signature Standard and the Key Exchange Algorithm (KEA) implemented on the FORTEZZA crypto card. In order to be used in a larger number of environments, the products should provide support for a wider range of public key algorithms, thus allowing users within specific environments to choose the algorithms that they will use. It should be noted that the RSA algorithm is unique in that it may be used for both digital signature and encryption/key exchange purposes. Other public key algorithms may only be used for one of these applications.

Strong client authentication needs to be supported. Current implementations of the SSL and S-HTTP protocols are focused on transmission security. Only servers have public-private key pairs and certificates. The impetus behind this seems to be a driving push to get credit cards across the Internet safely. This may be a worthy goal; however, it must be understood that if these products are to support all types of electronic commerce which includes the interchange of ideas, opinions, or sentiments as well as the buying/selling of commodities, support for client keys and certificates is needed. Given that the Internet has been used more for the former than it has for the latter, it seems that security solutions providing protection for ideas or information would be at least as sought after as those that provide security for the transmission of credit card numbers.

Products should include an interface to an X.500 Directory. In many public key management approaches, certificates are stored in an X.500 Directory, which is a publicly-accessible place for storing information. Directory servers play a pivotal role in any large scale public key infrastructure where key material must be disseminated to large numbers of people and applications operating in various geographic locations.

Certificate revocation needs to be supported. When client certificates are supported, servers will need to know that the clients to which they are speaking are genuine clients and not a masquerading client. Adding support for certificate revocation means that the products will need to support Certificate Revocation Lists (CRLs), be able to retrieve CRLs and/or receive CRLs, and to process the information contained within them. CRLs are often stored in an X.500 Directory, providing another motivation for adding an X.500 Directory interface to the products so that CRLs may be retrieved from the Directory.

And, of course, more work needs to be done in the community to solve the issues associated with public key and certificate management. Competing approaches to public key management along with the need to support different public key algorithms have resulted in several different public key infrastructures being in existence today. It is unclear whether a single public key infrastructure that services the needs of all communities will exist in the future. If not, interoperability issues between different infrastructures will need to be resolved in the future.

Where do we go from here?

MITRE plans to continue with its cryptographic endeavors and its quest to integrate electronic commerce products with its security infrastructures. We believe this is an achievable goal. In the last few months, we have seen much more emphasis in the community on providing the services we need. Services such as strong client authentication, support for CRLs, and interfaces to external systems are receiving much more attention of late.

MITRE participates as a member of many standards' bodies and is actively involved in the World Wide Web Consortium (W3C). We also actively participate in the W3C security workshops. In addition, MITRE is hosting a Public Key Infrastructure (PKI) Invitational Workshop along with The National Institute of Standards and Technology (NIST) and the NII Security Infrastructure Program Management Office (SI-PMO) in September 1995 to resolve PKI interoperability technical issues. We are working with both vendors and customers to ensure that our needs and theirs are well understood within the community. We hope that these efforts, as well as the efforts of the so many others that have similar goals and desires, will create a truly open and interoperable world for electronic commerce.

Bibliography

Berkovits, Dr. S., Dr. S. Chokhani, J. A. Furlong, J. A. Geiter, and J. C. Guild, April 1994, *Public Key Infrastructure Study Final Report*, MTR 94W0000180, The MITRE Corporation, McLean, VA.

CCITT, Geneva, 1989, *Data Communication Networks: Directory*, Recommendation X.500 - X.521, Blue Book, Volume VIII–Fascicle VIII.8.

Federal Information Processing Standards Publication, Digital Signature Standard (DSS), 19 May 1994, FIPS PUB 186, U. S. Department of Commerce/National Institute of Standards and Technology, Gaithersburg, MD.

Federal Information Processing Standards Publication, Secure Hash Standard (SHS), 31 May 1994, FIPS PUB 180-1, U. S. Department of Commerce/National Institute of Standards and Technology, Gaithersburg, MD.

Heckman, Donald, 18 August 1993, *MOSAIC Key Management Concept*, Revision 2.4, National Security Agency, Ft. Meade, MD.

Hickman, K., and T. ElGamal, June 1995, *The SSL Protocol*, Internet Draft, IAB.

Kaliski, B., 3 June 1993, *An Overview of the PKCS Standards*, RSA Data Security, Inc.

Kent, S., February 1993, *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, RFC 1422, IAB IRTF PSRG, IETF PEM WG.

Kohl, J., *The Kerberos Network Authentication Service (V5)*, September 1993, RFC 1510, IAB IETF Network Working Group.

Public Key Cryptographic Standard (PKCS) #10: Certification Request Syntax Standard, 1 November 1995, Version 1.0, RSA Laboratories.

Rescorla, E., and A. Schiffman, December 1994, *The Secure HyperText Transfer Protocol*, Internet Draft, IAB.

Rescorla, E., and A. Schiffman, July 1995, *The Secure HyperText Transfer Protocol*, Internet Draft, IAB IETF Web Transaction Security Working Group.

Rivest, R. L., A. Shamir, and L. Adleman, February 1978, "On Digital Signatures and Public Key Cryptosystems," *Communication of the ACM*, Vol. 21, No. 2, pp. 120-126.

Rivest, R., April 1992, *The MD5 Message-Digest Algorithm*, RFC 1321, IAB IETF Network Working Group.

Steiner, J., C. Neuman, and J. Schiller, 1988, *Kerberos: An Authentication Service for Open Network Systems*, Paper presented at USENIX, Dallas, Texas.

Generic Extensions of WWW Browsers

Ralf Hauser Michael Steiner
Information Technology Solutions Department, IBM Research Division
Zürich Research Laboratory, CH-8803 Rüschlikon, Switzerland
tel: +41.1.724-8426, fax: +41.1.710-3608, email: {rah, sti}@zurich.ibm.com

Extended Abstract

August 10, 1995

Abstract

Current WWW browsers provide two main services: communication and information rendering. While this is sufficient for some purposes, many future applications will need more sophisticated processing on the user side before server-data can be presented to the user or before the user input can be transferred to the servers. For example, electronic payments ought to be seamlessly integrated into a customer's browser to enable him to shop via the Web. This paper proposes two pragmatic approaches to fulfill these requirements without altering current browser technology. We conclude with the proposal of a generalized extension framework for WWW browsers.

1 Introduction

The World Wide Web (WWW) has grown extremely fast in the past months, not only quantitatively but also qualitatively. New services are being added on a day-by-day basis and range from telephone directories to movie databases and "shopping malls". In particular, the "POST"-Method of HTTP [1] and the "FORMS"-feature of HTML[2] have made the WWW technology suitable to most Internet applications.

With new commercial applications emerging daily, it is impossible to anticipate all future needs and build a single browser that fulfills all requirements. One example for this are security requirements: the simple authentication mechanisms of early HTTP versions are no longer sufficient. Advanced services such as secure payments and intellectual property rights protection require new mechanisms based on multi-party security.

Therefore, browsers should provide interfaces to allow the easy addition of arbitrary extensions. Most current browsers support the concept of *external viewers* based on Internet Media (MIME) Types [3]. The users of a WWW browser can freely configure their preferred viewers, enabling them to receive an open-ended set of document types with this feature. However, permitting only external viewers as extensions is not sufficient. Services such as confidentiality, integrity protection and packetizing data are only intermediate processing steps. They are *filters* and need to return output back to the browser controlling the overall application run.

WWW-browsers are also proposed as exclusive user interfaces[1] or as HTTP engines. The browser is then the slave of a different application. This cannot be handled by the MIME extension feature, and the different browser implementations address this issue of "remote control" in different ways.

In the next section we shows how rudimentary filter extensions can be implemented with existing technology and virtually any browser without modifications of either source or object code. We use the example of a secure purchase and payment of a digital movie to illustrate the approaches. In section 3 we outline a generic framework that accommodates both filters as well as remote control.

[1] Shah [4] envisions using the WWW system as operating environment and the browsers as generic user interfaces also to other applications.

2 Short-Term Approaches

For short- and mid-term solutions, an extension mechanism must not require changes of the browsers but should depend only on the features implemented by the majority of available browsers.

We propose two pragmatic approaches with minimal dependencies on implementation specifics to "snap-in" a client-side extension into today's browsers . The first approach relies only on external viewers that use a special MIME type, and the second approach is based on a small daemon resembling the HTTP server (`httpd`) running on the user's local machine.

As an example to demonstrate our approaches we take the scenario that a User likes to purchase a movie from a server using a secure payment scheme (e.g *iKP*[5]).

2.1 External Viewer Approach

Figure 1 shows the purchase of a movie based on external viewers. This minimal approach consists of

1. a payment "extension" MIME viewer entailing
   - user interface routines to obtain the user's confirmation to join the purchase contract, and the
   - cryptographic mechanisms to build a secure digital credit cart slip.
2. a method to "re-integrate" the extension's output into the WWW browser.

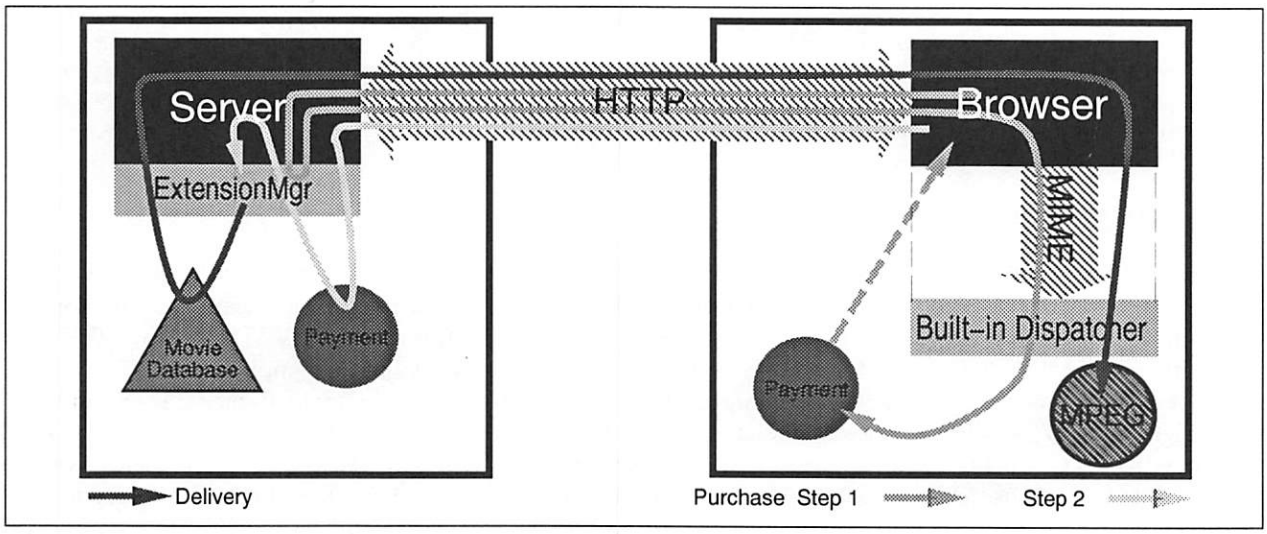


Figure 1: Payment of a Movie based on external viewers

Re 1: Whenever some input requires further processing on the user side, the server tags it with a specific MIME-type and submits it to the browser[2]. In our example of the purchase of a digital movie, the browser transparently passes this information to a "payment" extension in the form of a MIME viewer. The extension then displays contract parameters such as duration, number of frames per second, resolution, and price. Then it prompts the user for a confirmation. For small amounts, this confirmation can be provided automatically by the user's device. For larger amounts, the payment extension's policy will require the user to explicitly enter a PIN or a password to unlock the user's credit card for the generation of a digital credit-card slip.

Re 2: Such a re-integration of the information produced for a payment meta-protocol aims to feed data back to the browser or at employing the HTTP communication infrastructure that is already in place from

[2]For example, the user has filled out an order form and passed it to the shop's server. A first plausibility check, for example, whether the required amount is on stock or whether it requires extra freight negotiations, can already be performed by the server before turning the information of this flow into the first flow of the payment protocol under the pertinent special MIME type.

the browser to the server. In this way the payment extensions to not have to care about communication and can be unaware of potential gateways to cross on the way to the merchant. The first version of remote control features of both Mosaic[3] and Netscape[4] does not permit external applications to transmit more information transparently to a server via a browser than can be fitted into a URL. Most implementations expect this URL string to be only a few hundred characters long. For many situations this length is insufficient (e.g., with RSA a simple signed or encrypted block would already require 170 bytes).

One intermediate solution is to include the digital slip in a hidden input field of an HTML form that can be arbitrarily long. The entire html document containing this form is then deposited in a local file with a standard name. This file can be brought back into the browser in two ways:

- The last action of the payment extension is to cause the browser to load this local file by the aforementioned remote-control features. The user must then click *one* more time to send the form (containing the slip) to the server.
- If the browser entails no remote control feature, the server already provides a link to the standard local filename when supplying the user with the last information rendered by the HTML viewer. In this case, the last action of the browser is to inform the user that the local file is ready and the pertinent link can be clicked. This approach even requires *two extra user interactions* which will be unnecessary in implementations of long-term frameworks.

Evaluation

Such a short-term solution allows one to "**snap**" almost arbitrary extensions "**in**"to existing browsers. The requirements for the communication and operating environments are minimal. The solution, however, has two drawbacks. First, the filtering process sporadically needs user input to continue operation - input that is unnecessary from a logical point of view. Second, post-processing filters (i.e. filters that are employed after the last logically necessary user input) require an extra communication exchange with the server because the MIME dispatcher of the browser only operates on incoming data. This extra communication exchange is not possible with certain extensions[5].

2.2 Client-Side HTTP Daemon

The second short-term approach is to have an `httpd`-like daemon on the user's machine as shown in Figure 2. For security reasons, this extension `httpd` only accepts connections on a well-defined port from browsers on the same machine. When the payment for an electronic purchase is about to take place, the user clicks on a URL (provided by the shop's server) and causes a normal *POST* request to be sent to this local `httpd` (e.g.: `http://localhost:54321/buy`). The body of this request contains the information to assemble the digital credit card slip. To obtain further user input (such as credit card numbers or the sensitive PIN for transaction approval) this extension `httpd` either directly pops up with a special window for user keyboard input, or collects this information through the browser by sending back an appropriate HTML form.

There are two approaches for the extension `httpd` to transmit the final order with the digital slip to the server:

1. It puts the complete order information into a URL addressed to the shop's server. It then sends this URL with a *HTTP Redirect directive* to the browser. The browser therefore automatically issues a normal HTTP request with that URL to the server of the shop.
2. It acts as a *Proxy*; therefore, it contacts the shop's server directly and forwards the shop's response to the browser.

The main problem with this approach is how the shop's server can find out the port number of the extension `httpd` to construct the mentioned local URL. The answer is three-fold:

[3] http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/remote-control.html

[4] http://home.netscape.com/info/APIs/x-remote.html

[5] For example, communication-oriented post-processing such as packetizing bulk data, because the `GET` and `POST` method remain the only available communication primitives.

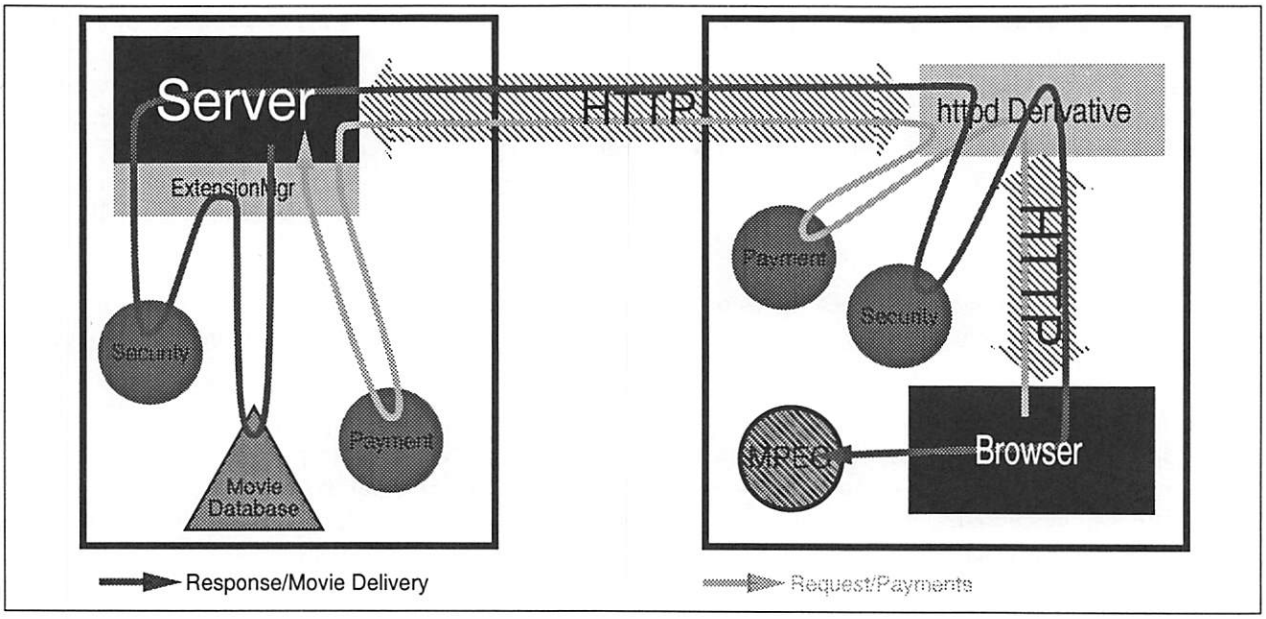


Figure 2: Payment with a Client-Side Extension HTTP-Daemon acting as a Proxy

- A well-known port will be established for the most common case.
- If the user cannot use this port for any reason, the port will be published as a special MIME-type of the form `extension-mgr/port-number`, where port-number stands for the port actually used. This will be sent with the HTTP *ACCEPT* header-field to the server and can be used to generate the URLs.
- If all this fails, the user will have to manually fill out a form that explicitly asks for the extension's port number, and then send it to the server.

Evaluation

The advantage is that this approach requires no extra user interaction to keep the filtering process alive. Additionally state can be kept easily between different communication exchanges.

Order transmission approach (1) has the advantage that the extension `httpd` only interacts with the user's browser and, therefore, can be totally shielded from outside communication. On the other hand we have to consider the limitation mentioned before of the amount of information that can be fitted into a URL.

Order transmission approach (2) does not suffer from such a size restriction. It can additionally react on faults as timeouts (e.g by sending retries) and pre-process the response of the shop's server. Figure 2 shows how the security extension could provide copyright protection by decrypting the delivered digital movie. The main disadvantage of this approach is that the daemon must be aware of proxies or gateways (e.g SOCKS) to always be able to reach the shop.

3 General Client Side Extension Framework

The problems of our approaches presented in the last section led us to look for a long term solution. This solution should also support certain extensions as load balancing by redirecting certain URLs to local files or geographically closer servers and outlining of document structures ([6]) cannot be handled easily by the approaches mentioned before. We can relax our requirement of not changing browsers. Therefore we propose a change in the architecture of browsers.

In the currently popular WWW browsers, the HTML viewer, the HTTP communications module, and a primitive "MIME document dispatcher" are combined in one monolithic package. Some implementations add some limited capabilities of handling remote control events as well.

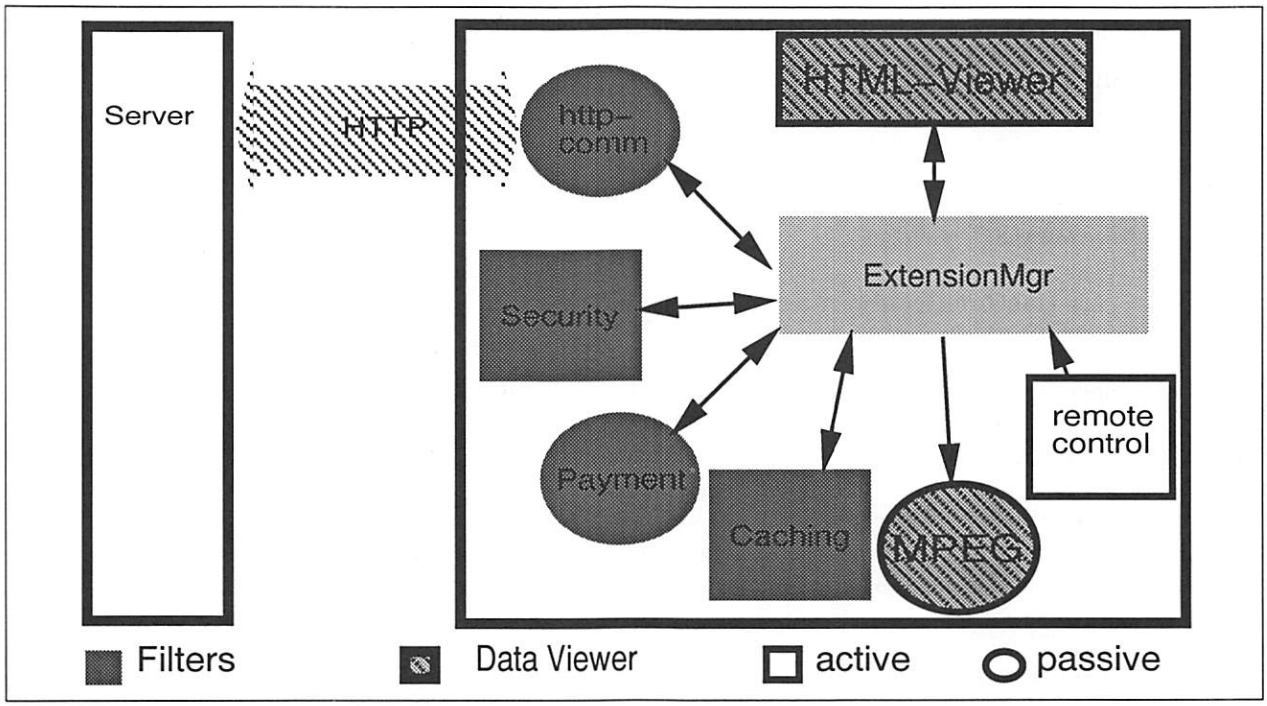


Figure 3: Examples of current Extensions

This architecture can be greatly simplified and its flexibility can be increased if we view the browser as a generic *extension manager* who acts merely as a dispatcher. Not only MIME extensions, but also the communication toolbox and the HTML viewer are subsumed as extensions.

This results in an architecture with only three logical components:

- the **extension manager**,
- **passive** extensions.
- **active** extensions.

Passive extensions are called by the extension manager. Their purpose is to either filter and transform data, to serve as gateways to different protocols or to render data.

Active extensions control the execution of the extension manager and initiate "web-actions". Note that the two types of extensions are not mutually exclusive. The HTML-viewer is clearly a data viewer but when it reacts on user interaction and issues a request it can also be considered as an active extension.

This leads to the following sketch of a minimal interface defining the interaction between the extension manager and extensions/external applications.

The extension manager exports two functions for active extensions. Parameters named `request` or `response` are required to be valid HTTP requests rsp. responses:

- **ProcessURL(**
  `IN: request`
  `IN: returnMeResponse?`
  `OUT: [response if returnMeResponse?]`
  ),
- **RenderMIME(**
  `IN: response`
  `IN: associated URL`
  ),

The extension has similarly to provide one or both of the following two functions:

- **HandleURL(**
  `IN: request`
  `OUT: requestOrResponse`
  ),
- **HandleMIME(**
  `IN: response`
  `OUT: [requestOrResponse if not(IsRenderer?)]`
  ),

These functions are registered either statically with a configuration file or dynamically by calling specific registration functions of the extension manager. Together with the callback function a regular expression is registered. The extension manager tries it to match with requested URLs rsp. MIME-types to find out which extension to take. If matching extensions are found, the extension manager will take the best match and call the associated callback-function.

Once the call to a filter returns, the extension manager inspects `requestOrResponse` and depending on the type of the return value looks for the next MIME or URL filter to call. Data viewer do not return any data and the extension manager stops processing.

If an external application wants the resulting data of a `ProcessURL` [6] call regardless of its MIME type, it will set `returnMeResponse?` in the `ProcessURL` call. Instead of handing the final response to a data viewer according to the MIME-type matching rules the extension manager will return it to the caller.

3.1 Example

Our example begins with a user who has already browsed the price list of a movie-on-demand WWW shop and has made a choice. Therefore, the HTML viewer knows the URL of the selected digital movie as well as the desired payment protocol. The user's click is processed in the following steps:

1. The HTML viewer will initiate the operation by calling the extension manager with:

```
ProcessURL(
  ''POST 3KP://movie.com/ HTTP/1.0

   Merchandise=Entertainment/movie324.mpg,
   Value=$5,
   Shop-Name=MovieComInc,
   Shop-Certificate=...'',
   FALSE
   returnDataHandle
).
```

2. The extension manager looks for a registered extension and will match "3̂kp:*". It will call the payment extension then with:

```
HandleURL(
  ''POST 3KP://movie.com/ HTTP/1.0

   Merchandise=Entertainment/movie324.mpg,
   Value=$5,
   Shop-Name=MovieComInc,
   Shop-Certificate=...'',
   returnDataHandle
).
```

   The payment extension will generate a payment token and puts the following HTTP request in `returnDataHandle`:

[6] `ProcessURL` together with `RenderMIME` replace the current "remote-control" features of some browsers.

```
``POST http://movie.com/3kp-buy HTTP/1.0

Merchandise=Entertainment/movie324.mpg,
PayToken="XYZ" ''
```

Note that the payment extension may actively call the extension manager during its operation through the `ProcessURL` interface, for example, to retrieve some other party's public key.

3. Inspecting `returnDataHandle` the extension manager determines that it should call the HTTP communications module[7]. This module handles the HTTP request and returns the HTTP response with the requested document from the remote server in `requestOrResponse`:

```
``HTTP/1.0 200 OK
Date:  Friday, 23-Jun-95 13:52:57 GMT
MIME-version:  1.0
Content-type:  application/copyright_protect

..  encrypted data ...''
```

4. Because the movie-on-demand shop has some copyright protection in place, the responses MIME type will match the copyright protection extension and call the `HandleMIME` callback corresponding to this extension. In a simple case, this callback would decrypt the data with a session key and return the movie with MIME type MPEG.

5. Finally, the extension manager will hand the movie324.mpg data to the movie viewer (e.g. an MPEG viewer).

3.2 Remarks

The HTTP communications module need not be the only one capable of changing the parameters from HTTP requests to HTTP responses. A caching filter returns the document if it is already in the cache, and an access filter for a browser at a public place will return a document containing an error message if it refuses some requests that may infer high costs or are against the charter for which the terminal was provided. Protocol gateways (e.g., ftp, gopher, mailto) are also examples of filters that semantically convert HTTP requests into HTTP responses.

Response-side filters may also issue new requests, for example, for pre-fetching of referenced documents.

Integrity protection, confidentiality, and packetizing data for long messages require symmetrical filtering also on the server side. The Common Gateway Interface (CGI) [7] allows filtering on the server side but unfortunately, this interface only foresees one filtering step. To obtain multiple filtering steps, an extension manager can be placed within this single filter and emulate the flexibility and configurability of the client-side extension manager.

It is hoped that such a framework with standardized extension manager semantics and interfaces will allow arbitrary developers to offer their own, compatible and powerful extensions.

4 Comparison with Existing Approaches

In the final paper this section will discusses the current proposals, namely:

- Kristol's RFC for a HTTP extension mechanism [8],
- NCSA's *Common Client Interface (CCI)* [9], already available as a prototype,
- Spyglass's *Software Development Interface* [10],

[7] We call HTTP communications module the piece of code that implements the http semantics on a specific network infrastructure. Most current WWW browsers assume TCP/IP to be the infrastructure - in the future, this may change.

- W&A's API for WWW Applets[11],
- others.

These proposals are compared with our framework, and their shortcomings are analyzed.

5 Conclusions

This paper has presented two short-term solutions of how to **snap** almost arbitrary extensions **in**to current WWW browsers without altering their source code or executables. Our example illustrates how these two solutions enable electronic payments with acceptable user interfaces and highest security already with existing browser technology. We have built two prototypes that provided proof of these concepts.

The requirements for extra user input, which is logically not justified, or for peculiar "internal" browser inter-component communication is a sign that the technology originally was not designed for the use described here.

Therefore we have stated our own view of a sound architecture for flexibly configurable and composable WWW browsers of the future.

References

[1] Tim Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen. *Hypertext Transfer Protocol*, March 1995. Internet Draft, Expires September 8, 1995.

[2] HyperText Markup Language Specification - 2.0. Internet Draft, February 1995. Expires June 19, 1995.

[3] J. Postel. Media type registration procedure. Internet Request for Comment RFC 1590, March 1994.

[4] Rawn Shah. The World Wide Web System as an Operating Environment. http://www.rtd.com/people/rawn/os-paper.html, 1994.

[5] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, and Michael Waidner. iKP – a family of secure electronic payment protocols. In *First USENIX Workshop on Electronic Commerce*. USENIX, July 1995.

[6] Alan F. Slater. Extending W3 clients. In *Second International Conference on the World-Wide Web*, pages 899–908, Chicago, October 1994.

[7] Rob McCool. *The Common Gateway Interface*. NCSA, 1.1 edition, 1994.

[8] David M. Kristol. A proposed extension mechanism for HTTP. Internet Request for Comment RFC, January 1995.

[9] NCSA Mosaic Common Client Interface, September 1994. Version 1.0, http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/cci-spec.html.

[10] Paul Rohr. Software Development Interface, December 1994. http://www.spyglass.com/techreport/iapi.htm.

[11] Bert Bos. W&A an API for WWW applets, v1.1. http://www.let.rug.nl/ bert/W3A/W3A.html, February 1995.

Secure Coprocessors in Electronic Commerce Applications

Bennet Yee
Microsoft Corporation
Redmond, WA 98052
bsy@microsoft.com

J. D. Tygar
Carnegie Mellon University
Pittsburgh, PA 15213
tygar@cs.cmu.edu

Abstract

Many researchers believe electronic wallets (secure storage devices that maintain account balances) are the solution to electronic commerce challenges. This paper argues for a more powerful model — a secure coprocessor — that can run a small operating system, run application programs, and also keep secure storage for cryptographic keys and balance information.

We have built a system called Dyad, on top of a port of the Mach 3.0 microkernel to the IBM Citadel secure coprocessor. This paper describes the abstract architecture of Dyad and a general discussion of secure coprocessor implementations of a variety of electronic commerce applications:

- *Copy protection for software*
- *Electronic cash (including a critique of proposed solutions for point-of-sale electronic wallet systems)*
- *Electronic contracts*
- *Secure postage*

1 Introduction

Many researchers believe electronic wallets (secure storage devices that maintain account balances) are the solution to electronic commerce challenges [3, 31]. This paper argues for a more powerful model — a secure *coprocessor* — that can run a small operating system, run application programs, and also keep secure storage for cryptographic keys and balance information.

Secure coprocessors are tamper-proof sealed devices that have a processor, memory storage, and (optional) fast crypto-support. They are protected in that any attempt to penetrate them will result in all critical memory being erased.

Secure coprocessors have a number of advantages over electronic wallets. They have more powerful processors and larger amounts of memory, so they can do more: they can negotiate (and enforce) contracts that involve renting and redistributing intellectual information. If they incorporate a *secure display*, they can provide much greater protection for customers in point-of-sale applications. They can provide software copy protection, permitting much more general applications.

We have built a system called Dyad, on top of a port of the Mach 3.0 microkernel [19] to the IBM Citadel secure coprocessor [57]. This paper describes the abstract architecture of Dyad and a general discussion of secure coprocessor implementations of a variety of electronic commerce applications:

- Copy protection for software
- Electronic cash (including a critique of proposed solutions for point-of-sale electronic wallet systems)
- Electronic contracts
- Secure postage

A few years ago, only experimental prototypes of secure coprocessors existed, but today, the market is filling up. Manufacturers such as Cylink, IBM, National Semiconductor, Spyrus, Telequip, and others have announced secure coprocessor products. Several other major manufacturers will announce significant new products in the near future. There is a new FIPS standard for cryptographic modules (including secure coprocessors) [53] and

This work was supported in part by ARPA contracts F33615-90-C-1465 and F33615-93-1-1330, NSF Presidential Young Investigator Award CCR-9958087, matching funds from Motorola and TRW, a contract from the US Postal Service, and by an equipment grant from IBM. This work was done while the first author was at Carnegie Mellon University. This work is the opinion of the authors and does not necessarily represent the view of their employers, funding sponsors, or the US Government.

a new service to perform those evaluations [54]. This paper attempts to lay out the intellectual issues in the use of these new secure coprocessor products.

Because of length considerations, this paper does not discuss our Dyad implementation (see [60]), or additional applications of secure coprocessors arising from distributed computation (see [43, 44].)

2 Secure Coprocessor Model

A secure coprocessor is a hardware module containing (1) a CPU, (2) bootstrap ROM, and (3) secure non-volatile memory. This hardware module is physically shielded from penetration, and the I/O interface to the module is the only way to access the internal state of the module. (If the shield is somehow penetrated, then a secure coprocessor erases all critical memory.) This hardware module can store cryptographic keys without risk of release. More generally, the CPU can perform arbitrary computations (under control of the operating system); thus the hardware module, when added to a computer, becomes a true coprocessor. Often, the secure coprocessor will contain special purpose hardware in addition to the CPU and memory; for example, high speed encryption/decryption hardware may be used.

2.1 Security Properties of Secure Coprocessors

All security systems rely on a nucleus of assumptions. For example, it is often assumed that encryption systems are resistant to cryptanalysis. Similarly, we take as axiomatic that secure coprocessors provide private and tamper-proof memory and processing. These assumptions may be falsified: for example, attackers may exhaustively search cryptographic key spaces. Similarly, it may be possible to falsify our physical security axiom by expending enormous resources (possibly feasible for very large corporations or government agencies). We rely on a physical work-factor argument to justify our axiom, similar in spirit to intractability assumptions of cryptography. Our secure coprocessor model does not depend on the particular technology used to satisfy the work-factor assumption. Just as cryptographic schemes may be scaled or changed to increase the resources required to penetrate a cryptographic system, current security packaging techniques may be scaled or changed to increase the work-factor necessary to successfully bypass the secure coprocessor protections.

Secure coprocessors must be packaged so that physical attempts to gain access to the internal state of the coprocessor will result in resetting the state of the secure coprocessor (i.e., erasure of the secure non-volatile memory contents and CPU registers). An intruder might be able to break into a secure coprocessor and see how it is constructed; the intruder cannot, however, learn or change the internal state of the secure coprocessor except through normal I/O channels or by forcibly resetting the entire secure coprocessor. The guarantees about the privacy and integrity of the secure non-volatile memory provide the foundations needed to build distributed security systems.

2.2 Potential Platforms

Several physically secure processors exist, and more are forthcoming. Some of these are very secure, satisfying the highest security level specified by FIPS PUB 140-1 [53]. (This publication gives four security levels for cryptographic modules, including secure coprocessors.) Announced secure coprocessors include the μABYSS [55] and Citadel [57] systems from IBM, the iPower [34] encryption card by National Semiconductor, some extended implementations of the Clipper and Capstone systems [2, 51, 52] proposed by the NSA as DES replacements, the Crypta Plus [47] encryption card by Telequip, the CY512i [13] chip from Cylink, and various smartcard systems such as some GEMPlus or Mondex cards [22]. There will be additional announcements of systems with increased processing power from major vendors in the next few months. For a fuller descriptions of potential platforms, see [60].

3 Applications

Because secure coprocessors can *process* secrets as well as store them, they can do much more than just keep secrets confidential. We describe how to use secure coprocessors to realize exemplar electronic commerce applications: (1) copy protection, (2) electronic currency, (3) electronic contracts, and (4) secure postage meters. None of these are possible on physically exposed systems. These applications are discussed briefly below.

3.1 Copy Protection

Software is often charged on a per-CPU, per-site, or per-use basis. Software licenses usually prohibit making copies for use on unlicensed machines. This injunction against copying is technically unenforceable without a secure coprocessor. If the user can execute code on a physically accessible workstation, the user can also read that code. Even if attackers cannot read the workstation memory while it is running, we are implicitly depending on the

assumption that the workstation was booted correctly — verifying this property, as discussed in [60], requires the use of a secure coprocessor.

Software copy protection is complementary to electronic commerce. Without copy protection, rental or per-use charging of software is not possible. Here we discuss tradeoffs in using a secure coprocessor to implement software copy protection. (Dyad includes a software protection mechanism [60].)

3.1.1 Copy Protection with Secure Coprocessors

Secure coprocessors can protect executables from being copied and illegally used. The proprietary code to be protected — or at least some critical portion of it — is distributed and stored in encrypted form, so copying without the code decryption key is futile,[1] and this protected code runs only inside the secure coprocessor. Either public key or private key cryptography may be used to encrypt protected software. If private key cryptography is used, key management is still handled by public key cryptography. In particular, when a user pays for the use of a program, he sends the certificate of his secure coprocessor public key to the software vendor. This certificate is digitally signed by a key management center and is *prima facie* evidence that the public key is valid. The corresponding private key is stored only within the secure non-volatile memory of the secure coprocessor; thus, only the secure coprocessor will have full access to the proprietary software.

What if the code size is larger than the memory capacity of the secure coprocessor? We have two alternatives: we can *crypto-page* or we can split the code into protected and unprotected segments.

Section 4.3 discusses crypto-paging in greater detail, but the basic idea is to encrypt and decrypt virtual memory contents as they are copied between secure memory and external storage. When we run out of memory space on the coprocessor, we encrypt the data before it is flushed to unsecure external storage, maintaining privacy. Since good encryption chips are fast, we can encrypt and decrypt on the fly with little performance penalty.

Splitting the code is an alternative to crypto-paging. We can divide the code into a security-critical section and an unprotected section. The security-critical section is encrypted and runs only on the secure coprocessor. The unprotected section runs concurrently on the host. An adversary can copy the unprotected section, but if the division is done well, he or she will not be able to run the code without the secure portion. In μABYSS [56], White and Comerford show how such a partitioning should be done to maximize the difficulty of reverse engineering the secure portion of the application.[2]

Whether the proprietary code is split or not, the secure coprocessor runs a small security kernel. It provides the basic support necessary to communicate with the host or the host's I/O devices. With separate address spaces and a few communication primitives, the complexity of a security kernel can be kept low, providing greater assurance that a particular implementation is correct.

3.1.2 Previous Work

A more primitive version of the copy protection application for secure coprocessors appeared in [28, 56]; a secure-CPU approach using oblivious memory references (i.e., apparently random patterns of memory accesses) giving a poly-logarithmic slow down, appears in [18] and [35].

3.2 Electronic Currency

We have shown how to keep licensed proprietary software encrypted and allow only execute access. A natural application is to allow charging on a pay-per-use or metered basis. In addition to controlling access to the software according to the terms of a license, some mechanism must perform cost accounting, whether it tracks the number of times a program has run or tracks dollars in a user's account. More generally, this accounting software provides an *electronic currency* abstraction. Correctly implementing electronic currency requires that account data be protected against tampering — if we cannot guarantee integrity, attackers might be able to create electronic money

[1] Allowing the encrypted form of the code to be copied means that we can back up the workstation against disk failures. Even giving attackers access to the backup tapes will not release any of the proprietary code. (Note that our encryption function should be resistant to known-plaintext attacks, since executable binaries typically have standardized formats.) A more interesting question arises if the secure coprocessor may fail. Secure coprocessors may be used in a fault-tolerant fashion; see [60].

[2] We also examined a real application, gnu-emacs 19.22 [45], to show how it could be partitioned to run partially within a secure coprocessor. The X Windows display code should remain within the host for performance. Most of the emacs lisp interpreter (e.g., `bytecode.c`, `callint.c`, `eval.c`, `lread.c`, `marker.c`, etc) could be moved into the secure coprocessor and accessed as remote procedures. Any manipulation of host-side data — text buffer manipulation, lisp object traversal — required during remote procedure calls can be provided by a simple read-write interface (with caching) between the coprocessor and the host, with interpreter-private data such as catch/throw frames residing entirely within the secure coprocessor. Garbage collection does become a problem, since the garbage collector must be able to determine if a Lisp object is accessible from the call stack, a portion of which is inside the coprocessor. If we chose to hide the actions of the evaluator and keep the stack within the secure coprocessor hidden, this would require that the garbage collector code (`Fgarbage_collect` and its utilities) be moved within the secure coprocessor as well.

at will. Privacy, while perhaps less important here, is a property that users expect for their bank balance and wallet contents; similarly, electronic money account balances should also be private.

We argue that secure coprocessors can not only support electronic wallet functionality, but that they also offer stronger guarantees than existing and proposed electronic wallets. In particular, electronic coprocessors offer consumer protection unavailable with existing electronic wallets. (We have built an electronic currency mechanism on top of Dyad, see [60].)

3.2.1 Electronic Money Models

Several models can be adopted for handling electronic funds. Any implementation of these models should follow the standard transactional model, i.e., to group together operations in a *transaction* having these three properties [20, 21]:

1. *Failure atomicity.* If a transaction's work is interrupted by a failure, any partially completed results will be undone.

2. *Permanence.* If a transaction completes successfully, the result of its work will never be lost, except due to a catastrophic failure.

3. *Serializability.* Concurrent transactions may occur, but the results must be the same as if they executed serially. This means that temporary inconsistencies that occur inside a transaction are never visible to other transactions.

These transactional properties are requirements for the safe operation of any database, and they are absolutely necessary for any electronic money system.

In the following, we discuss various electronic money models, their security properties, and how they can be implemented using present day technology. (We have built an electronic currency system on top of Dyad.)

The first electronic money model is based on the cash analogy. In this model, electronic cash has similar properties to cash:

1. Exchanges of cash can be effectively anonymous.

2. Cash cannot be created or destroyed except by national treasuries.

3. Cash transfers require no online central authority.

(Note that these properties are actually stronger than that provided by real currency — serial numbers can be recorded to trace transactions. Similarly, currency can be destroyed.)

The second electronic money model is based on the credit cards/checks analogy. Electronic funds are not transferred directly; rather, promises of payment, cryptographically signed to prove authenticity, are transferred instead. A straightforward implementation of the credit card model fails to exhibit any of the three properties above. However, by applying cryptographic techniques, anonymity can be achieved in a cashier's-check-like scheme (e.g., Chaum's Digicash model [10], which lacks transactional properties such as failure atomicity — see section 3.2.3), but the latter two requirements (conservation of cash and no online central authority) remain insurmountable. Electronic checks must be signed and validated at central authorities (banks), and checks/credit payments en route "create" temporary money. Furthermore, potential reuse of cryptographically signed checks requires that the recipient must be able to validate the check with the central authority prior to committing to a transaction.

The third electronic money model is based on the bank rendezvous analogy. This model uses a centralized authority to authenticate all transactions and is poorly suited to large distributed applications. The bank is the sole arbiter of account balance information and can implement the access controls needed to ensure privacy and integrity of the data. Electronic Funds Transfer (EFT) services use this model — there are no access restrictions on deposits into accounts, so only the person who controls the source account needs to be authenticated.

We examine these models one by one.

With electronic currency, integrity of accounting data is crucial. We can establish a secure communication channel between two secure coprocessors by using a key exchange cryptographic protocol and thus use cryptography to maintain privacy when transferring funds. To ensure that electronic money is conserved (neither created nor destroyed), the transfer of funds should be failure atomic, i.e., the transaction must terminate in such a way as to either fail completely or fully succeed — transfer transactions cannot terminate with the source balance decremented without having incremented the destination balance or vice versa. By running a transaction protocol such as two-phase commit [7, 15, 58] on top of the secure channel, secure coprocessors can transfer electronic funds from one account to another in a safe manner, providing privacy and ensuring that money is conserved. Most transaction protocols need stable storage for transaction logging to enable the system to roll back when a transaction aborts. On large transaction systems this typically

has meant mirrored disks with uninterruptible power supplies. With the simple transactions needed for electronic currency, the per-transaction log typically is not that large, and the log can be truncated after transactions commit and further communications show all relevant parties have acknowledged the transaction. Because each secure coprocessor handles only a few users, small amounts of stable storage can satisfy logging needs. Furthermore, because secure coprocessors have secure non-volatile memory, we only need to reserve some of this memory for logging. The log, accounting data, and controlling code are all protected from modification by the secure coprocessor, so account data are safe from all attacks; their only threats are bugs and catastrophic failures. Of course, the system should be designed so that users should have little or no incentive to destroy secure coprocessors that they can access. This is natural when one's own balances are stored on a secure coprocessor, much like the cash in one's wallets.

If the secure coprocessor has insufficient memory to hold account data for all the users, the code and accounting database may be written to host memory or disk after obtaining a cryptographic checksum (see discussion of crypto-sealing in section 4.3). For the accounting data, encryption may alternatively be employed since privacy is usually also desired.

Note that this type of decentralized electronic currency is *not* appropriate for smartcards unless they can be made physically secure from attacks by their owners. Smartcards are only quasi-physically secure in that their privacy guarantees stem solely from their portability. Secrets may be stored within smartcards because their users can provide the physical security necessary. Malicious users, however, can violate smartcard integrity and insert false data.[3]

Secure coprocessor mediated electronic currency transfer is analogous to rights transfer (not to be confused with rights copying) in a capability-based protection system [59]. Using the electronic money — e.g., spending it when running a pay-per-use program — is analogous to the revocation of a capability. This type of model relies on the idea of secure-coprocessor-protected unforgeable electronic tokens. In addition to electronic money, these unforgeable tokens are useful for many other applications. Electronic tokens can be created and destroyed by a few trusted programs. For pay-per-use applications, the token is created by the vendor's sales program and destroyed by executing the application — the exact time of destruction of the token is a vendor design decision, since runs of application programs are not, in general, transactional in nature. However, the trusted electronic currency manager running in the secure coprocessorcan use distributed transactions to transfer money and other electronic tokens. Transaction messages are encrypted by the secure coprocessor's basic communication layer, providing privacy and integrity of communications. (Traffic analysis is beyond the scope of this work and is not addressed.)

What about the other models for handling electronic funds? With the credit card/check analogy, the authenticity of the promise of payment must be established. When the computer cannot keep secrets for users, there can be no authentication because nothing uniquely identifies users. Even if we assume that users can enter their passwords into a workstation without fear of their password being compromised, we are still faced with the problem of providing privacy and integrity guarantees for network communication. We have similar problems as in host-to-host authentication in that cryptographic keys need to be somehow exchanged. If communications are in plaintext, attackers may simply record a transfer of a promise of payment and replay it to temporarily create cash. While security systems such as Kerberos [46], if properly implemented [4], can help to authenticate entities and create session keys, they use a centralized server and have problems similar to those in the bank rendezvous model. While we can implement the credit card/check model using secure coprocessors, the inherent weaknesses of this model keep us from taking full advantage of the security properties provided by secure coprocessors; if we use the full power of the secure coprocessor model to properly authenticate users and verify their ability to pay (perhaps by locking funds into escrow), the resulting system would be equivalent to the cash model.

With the bank rendezvous model, a "bank" server supervises the transfer of funds. While it is easy to enforce the access controls on account data, this suffers from problems with non-scalability, loss of anonymity, and easy denial of service from excessive centralization.

Because every transaction must contact the bank server, access to the bank service will be a performance bottleneck. Banks do not scale well to large user bases. When a bank system grows from a single computer to several machines, distributed transaction systems techniques must be brought to bear in any case, so this model has no real advantage over the use of secure coprocessors in ease of implementation. Furthermore, if a bank's host becomes inaccessible, either maliciously or as a result of normal hardware failures, no agent can make use of any bank transfers. This model does not exhibit graceful degradation with system failures.

[3]Newer smartcards such as GEMPlus or Mondex cards [22] feature limited physical security protection, though the types of attacks these cards can withstand have not been published.

3.2.2 Point-of-Sale Terminals

In addition to their use in networked computers, secure coprocessors can be used for commercial transactions at point-of-sale terminals. For this application, we would need portable secure coprocessor form factors, such as smartcards or PCMCIA cards. Unlike the networked PC scenario where the users can be familiar with particular PCs they use, customers at a point-of-sale terminal have no reason to trust its integrity.

Point-of-sale use of secure coprocessors is vulnerable to a very important class of threats: communication spoofing between the secure coprocessor and the user. This problem arises because there is no private communications path [50] between the user and the secure coprocessor. A *secure display* only displays data to the user originating from the secure coprocessor and guarantees that the displayed data can not be tapped by a third party; such a display would provide secure one-way communication from the secure coprocessor and the user.

Today's smartcards and PCMCIA cards do not incorporate secure displays. Thus, for point-of-sale use, the user must rely on the display on the point-of-sale terminal to inform him of the total price. Unlike traditional paper credit-card-imprint slips, a secure coprocessor's digital signature is on a document that is never shown to the user — whatever per-signature user authorization required is performed blind, and the secure coprocessor might sign a purchase order for a $10,000 gold watch when the point-of-sale terminal is displaying "$1.98 watch batteries." Furthermore, to prevent a user authorization replay attack, some method for securely transferring the user authentication/authorization input to the secure coprocessor is required.

To permit secure input of user passwords to a secure coprocessor and to display purchase information, a secure display suffices: we use the secure display as a one-way secure channel over which we transmit a one-time pad, i.e., a cryptographically random string. The user then uses the point-of-sale terminal's keyboard (perhaps via arrow keys) to modify the displayed string into the user's password. For example, if your password was "SHOELACE" and the displayed string was "QZKNCFLX", you would press the [↑] arrow twice to change the "Q" to an "S", and then press the [→] arrow to advance to the next character, etc. (This is an idea adapted from [1].) Price information can be shown on a secure display in the obvious way.

Without a secure display of purchase data and secure entry of passwords, point-of-sale use of secure coprocessors does *not* increase the security of point-of-sale commerce over existing credit card systems. One much touted property of using smartcards in lieu of mag-stripe credit cards is customer non-repudiation and the elimination of merchant fraud. However, while the cryptographic signature keys may be secure, smartcards without some form of secure display can not link the signature to the purchase due to the absence of customer review. Thus customers are still vulnerable to merchant fraud — rather than modifying the numbers on a credit card slip after the fact, the merchant can simply introduce a difference between data presented to the user and the users' secure coprocessor.

3.2.3 Previous Work

An alternative to the secure coprocessor managed electronic currency is Chaum's Digicash protocol [8, 10]. In such systems, anonymity is paramount, and cryptographic techniques are used to preserve the secrecy of the users' identities. No physically secure hardware is used, except in the *observers* refinement to prevent double spending of electronic money (rather than detecting it after the fact).[4]

Chaum-style electronic currency schemes are characterized by two key protocols. The first is a *blind signature protocol* between a user and a central bank. During a withdrawal, the user obtains a cryptographically signed check that is probabilistically proven to contain an encoding of the user's identity. The user keeps the values used in constructing the check secret; they are used later in the spending protocol.

The second protocol is a randomized interactive protocol between a user and a merchant. The user sends the blind-signed check to the merchant and interactively proves that the check was constructed appropriately out of the secret values and reveals some, but not all, of those secrets. The merchant "deposits" to the central bank the blind-signed number and the protocol log as proof of payment. This interactive spending protocol has a flavor similar to zero-knowledge protocols in that the answers to the merchant's queries, if answered for both values of the random coin flips, reveal the user's identity. When double spending occurs, the central bank gets two logs for the same check, and from this identifies the double spender.

There are a number of problems with this approach. First, any system that provides complete anonymity is currently illegal in the United States, since any monetary transfer exceeding $10,000 must be reported to the government [12], employee payments must be reported similarly for tax purposes [11], stock transfers must be

[4] The observers model employs a physically secure hardware module to detect and prevent double spending. Chaum's protocol limits information flow to the observer, so that the user need not trust it to maintain privacy; however, it must be trusted to not destroy money. Secure coprocessors achieve the same goals with greater flexibility.

reported to the Securities and Exchange Commission, etc. Second, in a real internetworked environment, network addresses are required to establish and maintain a communication channel, barring the use of trusted anonymous forwarders — and such forwarding agents are still subject to traffic analysis. Providing real anonymity in the high level protocol is useless without taking network realities into account. Third, Chaum's cryptographic protocols do not handle failures, and any systems based on them cannot simultaneously have transactional properties and also maintain anonymity and security. A transaction abort in the blind signature protocol either leaves the user with a debited account and no electronic check or a free check. A transaction abort in the spending protocol either permits the user to falsify electronic cash if the random coin flips are reused when the transaction is reattempted (e.g., the network partition heals), or reveals identifying information to the merchant if new random coin flips are generated when the transaction is reattempted.

Clearly, to provide a realistic distributed electronic currency system, transactional properties must be provided. Unfortunately, the safety provided by transactions and the anonymity provided by cryptographic techniques appear to be inherently at odds with each other, and the tradeoffs made by Chaum-style electronic cash systems for anonymity instead of safety are inappropriate for real systems.

Another class of electronic money system is server-based. NetBill [42] is one type of such a system. NetBill implements the credit card model of electronic currency. A central server acts as a credit provider for users who can place a spending limit on each authorized transaction, and it provides billing services to the service providers. No true anonymity is achieved: the central server has a complete record of every user's purchases and the records for the current billing period is sent to users as part of their bill. Some scaling may be achieved through replication, but in this case providing hard credit limits require either distributed transactions, or every user must be assigned to a particular server, making the system non-fault tolerant.

Other approaches include anonymous credit cards [30] or anonymous message forwarders to protect against traffic analysis, at the cost of adding centralized servers back to the system.

3.3 Electronic Contracts

One of the most exciting applications of secure coprocessors is the use of electronic contracts. Electronic contracts are a natural extension to the "basic" electronic commerce approach. Where existing electronic commerce systems provide a basic, two-party contract which offered money for goods, a full electronic contract approach permits multi-party contracts, delegation, and a richer set of contractual primitives.

Electronic contracts provide enabling technology for creating electronic marketplaces [17]. Applications include the idea of superdistribution of software [33], and the creation of electronic futures markets.

In superdistribution, the idea is that the traditional software distribution channel is replaced by allowing a software buyer to resell the software on the manufacturer's behalf. When we look at this in the electronic contracts viewpoint, the customer is entering into a contract with the software manufacturer whereby the customer not only obtains the rights to use that software, but also the rights to make the same contract with other potential customers on the manufacturer's behalf. Such a self-replicating contract is a relatively simple three-party contract, where all of the contractual terms — electronic money transfer, rights to run a program, and making more electronic contracts — are enforceable by a secure coprocessor. See Figure 1 for an example superdistribution contract.

Having an expressive electronic contract language also enables the creation of electronic markets not previously possible. For example, air travel requirements — travel destination and approximate times — may be written up as an electronic contract containing the maximum price that the user is willing to pay, and this contract may be put up for auction. Travel agents bid for and buy the right (and obligation) to fulfill such contracts, increasing the efficiency of the travel market; additionally, travel agents may speculate on airline pricing and offer a higher bid in anticipation of fare reductions. Note, furthermore, that airline tickets may also be objects handled by the electronic contract system: these may simply be electronic documents signed by the airline giving the customer the right to travel on a particular flight, or even a token of a specific token type which permits travel on a certain flight.

In full generality, the objects referred to within electronic contracts will not always be objects that are managed by secure coprocessors, and this necessarily implies that external adjudication will be required when breaches of contracts occur. Furthermore, the user may not be able to satisfy the contractual demands, e.g., a broker who (speculatively) sells run-time on a mainframe may find all the cycles already allocated.

Our electronic contract model is built on the following two secure coprocessor-provided primitive objects: (1) unforgeable tokens and (2) computer-enforced contracts.

Unforgeable tokens are protected objects conserved by secure coprocessors; they are freely transferable, but can

```
software_distributor(signatory  id_t     manuf,
                     signatory  id_t     distributor,
                     key_t              sw_key,
                     int                manuf_profit,
                     id_t               prev_distr,
                     int                prev_distr_cut,
                     time_t             expire)
{
        int     price;

        terminates when
                date() >= expire;

access(none):
        super_buy(id_t          buyer,
                  money_t       cash @ buyer)
        {
                int  profit;
                     /* profit for this distributor; no Amway tree */
                if (cash->amount < price) reject;
                     /* cannot sell at a loss */
                xfer(cash,manuf->in_register,manuf_profit);
                xfer(cash,prev_distr->in_register,prev_distr_cut);
                profit = price - manuf_profit - prev_distr_cut;
                xfer(cash,distributor->in_register,profit);
                xfer(sw_key,buyer,1);
                software_distributor(manuf,buyer,sw_key,manuf_profit,
                                     distributor,profit,expire);
                     /* to do Amway, we would pass profit up
                      * the distr chain rather than all at once here */
        }
access(distributor):
        set_price(int           new_price)
        {
                /* pricing must at least pay for manufacturer profit */
                if (new_price < manuf_profit + prev_dist_cut) reject;
                price = new_price;
                enable_access(super_buy,all);
        }
}
```

Figure 1: Software Superdistribution Contract

In this example, the software retail distributor enters into a contract with a software manufacturer, which enables the distributor to sell the software to customers for customer use and at the same time permit the customer to redistribute the software under the same contractual terms. For the duration of the contract, the distributor gains the power to make new contracts on the manufacturer's behalf.

be created and destroyed only by the agents that issued them (or their designees). Furthermore, the transfer of tokens occur in a transactional manner, so that the number of tokens is a conserved quantity (excepting explicit action by their issuer).

Tokens are useful for representing electronic currency and execute-only rights to a piece of software (much as in capability systems). In the case of rights such as execute-only rights, the token provides access to cryptographic keys that may be used (only) within the secure coprocessors to run code. Electronic currency and execution rights are subtypes of tokens and inherit the transactional transfer property from tokens.

Contracts are another class of protected objects. They are created when two parties agree on a contract draft. Contracts contain binding clauses specifying actions that each of the parties must perform or actions that the secure coprocessors will enforce, along with "method" clauses that may be invoked by certain parties (not necessarily restricted to just the parties who agreed on the contract). Time-based clauses and other event-based clauses may also exist. Contractual obligations may force the transfer of tokens between parties.

Contract drafts are typically instantiated from a contract template. We can think of a contract template as a standardized contract with blanks which are filled in by the two parties involved, though certainly "custom" contracts are possible. Contract negotiation consists of an offerer sending a contract template along with the bindings (values with which to fill in blanks) to the offeree. The offeree either accepts or rejects the contract. If it is accepted, a contract instance is created whereby the contract bindings are permanent, and any immediate clauses are executed. If the draft is rejected, the offeree may take the contract template and re-instantiate a new draft with different bindings to create a counter-offer, whereupon the roles of offerer and offeree are reversed.

From the time that a contract is accepted until it terminates, the contract is an active object running in one or more secure coprocessors. Methods may be invoked by users or triggered by external events (messages from the host, timer expiration). The method clauses of a contract are access-controlled: they may be optionally invoked by only one party involved in the contract — or even by a third party who is under no contractual obligations.

Contractual clauses can require one of the parties to accept further contracts of certain types. One example of this is a requirement for action to be completed by a certain deadline, e.g., for a contractor to solve some problem or write some code before a project completion date. Another is a contract between a distributor and a software house, where the software house requires the distributor to accept sales contracts from users for upgrading a piece of software.

3.4 Secure Postage

While cryptographic methods have long been associated with mail (dating back to the use by Julius Caesar described in his book *The Gallic Wars* [9]), they have generally been used to protect the contents of a message, or in rare cases, the address on an envelope (protecting against traffic analysis). In this section, we examine the use of cryptographic techniques to protect the *stamp* on an envelope. (We are actively working with US Postal Service to define standards for the use of secure postage [23].)

The US Postal Service, with almost 40,000 autonomous post office facilities, handles an aggregate total of over 165 billion pieces of mail annually [40]. Most mail is metered or printed. (Figure 2 shows an example of a postage meter indicia.) Traditional postage meters must be presented to a branch post office to be loaded with postage. The postage credit is stored in a register sealed in the machine. As each letter is franked, the amount is deducted from the machine's credit register. Postal meters are subject to at least four types of attack: (1) the credit recorded in the postage meter may be tampered with, allowing the user to steal postage; (2) the postage meter indicia may be forged or copied; (3) a valid postage meter may be used by an unauthorized person; and (4) a postage meter may be stolen.[5]

With modern facilities for barcoding machine readable digital information, it would be easy to replace old-fashioned human readable indicia by indicia which are either entirely or partially machine readable. These indicia could encode a digitally signed message which would guarantee authenticity. If this digital information included unique data about the letter (such as the date mailed, zip codes of the originator and recipient, etc.), the digitally signed indicia could protect against forgery or copying. A rough outline of how such a system might work was detailed by Pastor [36].

Unfortunately, a digitally signed indicia may be vulnerable to additional types of attack:

1. If cryptographic systems are misused, the system may be directly attacked.

2. Even if cryptographic techniques are used correctly, if the adversary has physical access to the postage

[5] 82,000 postage meters in the U. S. are currently reported as lost or stolen [41].

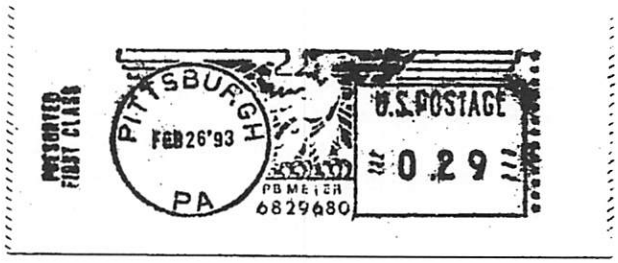


Figure 2: Postage Meter Indicia Can Be Easily Copied or Forged

meter, he may be able to tamper with the credit register.

3. Even if the credit is tamper-proof, a postage meter may be opened and examined to discover cryptographic keys, allowing the adversary to build new bogus postage meters.

4. The protection scheme may depend on a highly available network connecting post office facilities in a large distributed database. Since 40,000 autonomous post office facilities exist, such a network would suffer from frequent failures and partitions, creating windows of vulnerability (with 165 billion pieces of mail each year, a database to check the validity of digitally signed meter indicia appears infeasible.)

We outline a protocol for protecting cryptographic indicia, and demonstrate that the use of a secure coprocessor can address all of the above concerns. With the use of cryptography and secure coprocessors, it is possible to build a PC-based system that can produce fully secure postage indicia.

3.4.1 Cryptographic Indicia

A cryptographic postage indicia is an indicia that can demonstrate to the postal authorities that postage has been paid. Unlike the usual stamps purchased at a post office, these are printed by a conventional output device, such as a laser printer, directly onto an envelope or a package. Because such printed indicia can be copied, cryptographic and procedural techniques must be employed to minimize the probability of forgery.

We use cryptography to provide a crucial property: the indicia depends on the address. A malicious user may copy a cryptographic indicia, but any attempts to *modify* it or the envelope address will be detected. To achieve this goal, we encrypt (or cryptographically checksum) as part of the indicia information relevant to the delivery of the particular piece of mail — e.g., the return address and the destination address, the postage amount, and class of mail, etc, as well as other identifying information, such as the serial number of the software instance producing the indicia, a sequence number for the indicia, and the date/time (a *timestamp*). The information, including the cryptographic signature or checksum, is put into a barcode. The barcode must be easily printable by commodity or after-market laser printers, it must be easily scanned and re-digitized at a post office, and it must have sufficient information density to encode all the bits of the indicia on the envelope within a reasonable amount of space. Symbol Technologies' PDF417 [26, 37, 38], for example, can encode 400 bytes per square inch, sufficient for cryptographic indicia. Figure 3 shows an example of PDF417's density.

Six lines of 40 full ASCII characters for each address,[6] four bytes each for hierarchical authorization number, a serial number for the software instance that produced the indicia, the indicia sequence number, the postage/class, and the time, totals to under 500 bytes of data. (Using PDF417, 500 bytes takes 1.24 square inches.)

The cryptographic signature within the indicia prevents many forms of replay attacks. Malicious users will not find it useful to copy the indicia, since the cryptographic signature prevents them from modifying the indicia to change the destination addresses, etc, so the copied indicia may only be used to send more mail to the same destination address. If duplicate detection is used (see below) then even this threat vanishes. The timestamps and serial numbers also limit the scope of the attack by restricting the lifetime of copies and permitting law enforcement to trace the source of the attack.

Because cryptographic indicia also includes source in-

[6] Instead of a 40 character address, a 11-digit extended ZIP code presently in use internally by the U. S. Postal Service may be used instead; an eleven digit address fits in 37 bits.

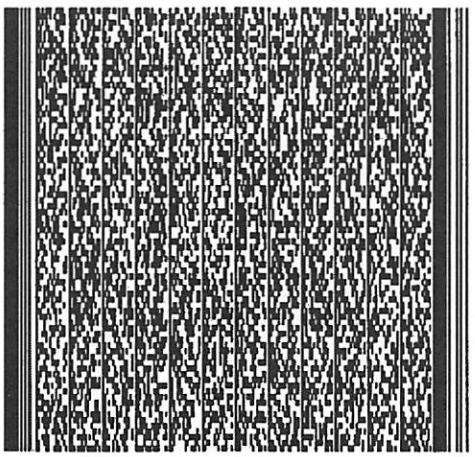


Figure 3: PDF417 encoding of Abraham Lincoln's Gettysburg Address

formation, a serial number, and the return address, duplicated indicia can also be detected in a distributed manner. Replays are detected by logging recent, unexpired indicia from processed mail. If the post office finds a piece of mail with a duplicate indicia, they will know that some form of forgery has occurred. We examine the practicality of replay detection next.

3.4.2 Detecting Replays

With a kilobyte of data per indicia, it would seem at first that replay detection is infeasible because of size of the database required. However, we can exploit the distributed nature of mail delivery and sorting.

The US Postal Service sorts mail twice. First, mail is sorted by destination zip code at a site near the source. Then, the mail is delivered (in large batches) to a site associated with the destination zip code, where the mail is again sorted, this time by carrier route. Every piece of mail destined for the same address passes through the same secondary sorting site, making it a natural place for detecting replays.

Detecting replays locally is feasible with today's technology. Using the 1992 figures of 165 billion pieces of mail per year handled at 600 regional sorting sites, with the simplifying assumption that the volume of mail is evenly distributed among these regional offices, we can obtain an estimate of the storage resources required. Assuming that cryptographic indicia expire six months after printing,[7] an average regional office will see approximately 130,000,000 indicia out of a national total of 80,000,000,000 indicia. If we store one kilobyte of information per indicia (doubling the above estimate) and assume that the entire current mail volume uses cryptographic indicia, this would require only 130 gigabytes of disk storage per facility for logging, well within the capacity of a single disk array system. The indicia database can be viewed as a sparse boolean matrix indexed in one dimension by software instance serial number and in the second dimension by indicia sequence number for that software instance.

To make replay detection even easier, we exploit the physical locality property: pieces of mail franked by a single device are likely to enter the mail processing system at the same primary sorting site. Therefore, cryptographic indicia from the same device are very likely to be canceled at the same regional office, and we can detect replays there. If any cryptographically franked piece of mail is sent from a different mail cancellation site, network connections can be used for real-time remote access of cancellation databases, or batch processing media such as computer tapes may be used. In the case of real-time cancellation, the network bandwidth required depends on the probability of the occurrence of such multi-cancellation-site processing, and on how quickly we need to detect replays. The canceled indicia database at each regional office need not be large — each device can simply encrypt a counter value in its indicia. We need only fast access to a bit vector of recently used, unexpired indicia counter values. These bit vectors are indexed by the device's serial

[7] The U. S. Postal Service claims to deliver more than 90% of all first class mail in three days, and more than 99% in seven days. Six months would appear to be a generous bound for mail delivery.

number and can be compressed by run-length encoding or other techniques. Only when a replay is detected might we need access to the full routing information.

4 System Architecture

We have implemented Dyad, a prototype secure coprocessor system. The Dyad architecture is based on operational requirements arising from the security applications in section 3. This section discusses Dyad's abstract system architecture based on the operational requirements during system initialization and during normal, steady state operation. A more detailed discussion of the concrete system architecture may be found in [60].

4.1 Operational Requirements

We begin by examining how a secure coprocessor interacts with the host during system boot and then describe system services that a secure coprocessor provide to the host operating system and user software.

To be sure that a system is securely booted, the bootstrap process must involve secure hardware. Depending on the host hardware (e.g., whether a secure coprocessor could halt the boot process in case of an anomaly) we may need secure boot ROM. Either the system's address space is configured so the secure coprocessor provides the boot vector and the boot code directly; or the boot ROM is a piece of secure hardware. In either case, a secure coprocessor verifies system software (operating system kernel, system related user-level software) by checking the softwares' signatures against known values. To check that the version of the software present in external, unsecure, non-volatile store (disk) is the same as that installed by a trusted party. Note that this interaction has the same problems faced by two hosts communicating via a unsecure network: if an attacker can completely emulate the interaction that the secure coprocessor has with a normal host system, it is impossible for the secure coprocessor to detect this. With secure coprocessor/host interaction, we can make very few assumptions about the host (it can not keep cryptographic keys). The best that we can do is to assume that the cost of completely emulating the host at boot time is prohibitively expensive.

The secure coprocessor ensures that the system securely boots; after booting, a secure coprocessor aids the host operating system by providing security functions. A secure coprocessor does not enforce the host system's security policy — this is the job of the host operating system. Since we know from the secure boot procedure that a correct operating system is running, we may rely on the host to enforce policy. When the host system is up and running, a secure coprocessor provides various security services to the host operating system:

- integrity verification of any stored data (by secure checksums);
- data encryption to boost storage media natural security; and
- encrypted communication channels (key exchange, authentication, private key encryption, etc).[8]

4.2 Secure Coprocessor Architecture

The boot procedure described above made assumptions about secure coprocessor capabilities. We refine the requirements for secure coprocessor software and hardware.

To verify that the system software is the correct version, the secure coprocessor must have secure memory to store checksums or other data. If keyless cryptography checksums such as MD5 [39], multi-round Snefru [32], or IBM's MDC [25] are one-way hash functions, then the only requirement is that the memory be protected from unauthorized writes. Otherwise, we must use keyed cryptographic checksums such as Karp and Rabin's technique of *fingerprinting* (see [27]). The latter approach requires that memory also be protected against read access, since both the hash value and the key must be secret. Similarly, cryptographic operations such as authentication, key exchange, and secret key encryption all require secrets to be kept. Thus a secure coprocessor must have memory inaccessible to all entities except the secure coprocessor itself — enough private non-volatile memory to store the secrets, plus private (possibly volatile) memory for intermediate calculations in running protocols.

How much private non-volatile and volatile scratch memory is enough? How fast must the secure coprocessor be to have good performance with cryptographic algorithms? There are a number of architectural tradeoffs for a secure coprocessor, the crucial dimensions being processor speed and memory size. They together determine the class of cryptographic algorithms that are practical.

4.3 Crypto-paging and Sealing

Crypto-paging is another technique for trading off memory for speed. A secure coprocessor encrypts its virtual

[8] Presumably remote hosts will also contain a secure coprocessor, though everything will work fine as long as remote hosts follow the appropriate protocols.

memory contents before paging it out to the host's physical memory (and perhaps eventually to an external disk), ensuring privacy. We need only enough private memory for an encryption key and a data cache, plus enough memory to perform the encryption if no encryption hardware is present. To ensure integrity, virtual memory contents may be *crypto-sealed* by computing cryptographic checksums prior to paging out and verifying them when paging in.

Crypto-paging and sealing are analogous to paging of virtual memory to disk, except for different cost coefficients. Well-known analysis techniques can be used to tune such a system [29, 61]. The cost variance will likely lead to new tradeoffs: computing cryptographic checksums is faster than encryption, so providing integrity alone is less expensive than providing privacy as well. On the other hand, if the computation can reside entirely on a secure coprocessor, both privacy and integrity can be provided for free.

Crypto-paging is a special case of a more general speed/memory trade off for secure coprocessors. We observed in [48, 49] that Karp-Rabin fingerprinting can be sped up by about 25% on an IBM RT/APC with a 256-fold table-size increase; when implemented in assembler on an i386SX, the speedup is greater (about 80%; see [60]). Intermediate-size tables yield intermediate speedups at a slightly higher increase in code size. Similar tradeoffs can be found for software implementations of DES.

4.4 Secure Coprocessor Software

A small, simple security kernel is needed for the secure coprocessor. What makes Dyad's kernel different from other security kernels is its partitioned system structure.

Like normal workstation (host) kernels, the secure coprocessor kernel must provide separate address spaces for vendor and user code in the secure coprocessor — even if we implicitly trust vendor and user code, providing separate address spaces helps isolate the effects of programming errors. Unlike the host's kernel, many services are not required: terminal, network, disk, and most other device drivers need not be part of the secure coprocessor. Indeed, since both the network and disk drives are susceptible to tampering, requiring their drivers to reside in the secure coprocessor's kernel is overkill — network and file system services from secure coprocessor tasks can be forwarded to the host kernel for processing. Normal operating system daemons such as printer service, electronic mail, etc. are entirely inappropriate in a secure coprocessor.

The only services crucial to the operation of the secure coprocessor are (1) secure coprocessor resource management; (2) communications; (3) key management; and (4) encryption services. *Resource management* includes task allocation and scheduling, virtual memory allocation and paging, and allocation of communication ports. *Communications* include both communication among secure coprocessor tasks and communication to host tasks; it is by communicating with host system tasks that proxy services are obtained. *Key management* includes management of authentication secrets, cryptographic keys, and system fingerprints of executables and data. With the limited number of services needed, we can easily envision using a microkernel such as Mach 3.0 [19], the NT executive [14], or QNX [24]. We only need to add a communications server and include a key management service to manage secure non-volatile key memory. If the kernel is small, we have more confidence that it can be debugged and verified. (In Dyad, we ported Mach 3.0 to the Citadel secure coprocessor.)

4.5 Key Management

Key management is a core portion of the secure coprocessor software. Authentication, key management, fingerprints, and encryption protect the integrity of the secure coprocessor software and the secrecy of private data. The bootstrap loader, in ROM or in secure non-volatile memory, controls the bootstrap process of the secure coprocessor itself. In the same way that the host-side bootstrapping process verifies the host-side kernel and system software, this loader verifies the secure coprocessor kernel before transferring control to it.

The system fingerprints needed for checking system integrity reside entirely in secure non-volatile memory or are protected by encryption while in external storage. (Decryption keys reside solely in secure non-volatile memory.) If the latter approach is chosen, new private keys must be selected for every new release of system software[9] to prevent replay attacks where old, buggy, secure coprocessor software is reintroduced into the system. Depending on the algorithm, storage of the fingerprint information can requires only integrity or both integrity and secrecy.

Other protected data held in secure non-volatile memory include administrative authentication information needed to update the secure coprocessor software. We assume that a security administrator is authorized to upgrade secure coprocessor software. The authentication data for the administrator can be updated along with the rest of the secure coprocessor system software; in either case, the

[9] One way is to use a cryptographically secure pseudo-random number generator [5, 6] with its internal state entirely in secure non-volatile memory.

upgrade must appear transactional, that is, it must have the properties of *permanence* (results of completed transactions are never lost), *serializability* (there is a sequential, non-overlapping view of the transactions), and *failure atomicity* (transactions either complete or fail such that any partial results are undone [16, 20, 21]). Non-volatile memory gives us permanence; serializability, while important for multi-threaded applications, can be enforced by permitting only a single upgrade operation at a time (this is an infrequent operation and does not require concurrency); and the failure atomicity guarantee can be provided as long as the secure non-volatile memory subsystem provides an atomic store operation. Update transactions need not be distributed nor nested; this simplifies the implementation.

5 For Further Information

Dyad allows a much broader class of electronic commerce activities than more narrowly defined electronic wallet systems. By using a secure coprocessor model, we are able to add substantial functionality.

Information on secure coprocessors and the Dyad implementation can be found on our WWW page: *http://www.cs.cmu.edu/afs/cs/project/dyad/www/*. Please send any email or inquiries for information to `dyad@cs.cmu.edu`.

References

[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. Technical Report 67, DEC Systems Research Center, October 1990.

[2] R. G. Andersen. The destiny of DES. *Datamation*, 33(5), March 1987.

[3] Ross Anderson. Making smartcard systems robust. In *IFIPS First Smartcard Research and Advanced Application Conference*, pages 1–14, Lille, France, October 1994.

[4] S. M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. Submitted to *Computer Communication Review*, 1990.

[5] Blum, Blum, and Shub. Comparison of two pseudo-random number generators. *Advances in Cryptology: CRYPTO-82*, pages 61–79, 1983.

[6] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[7] Andrea J. Borr. Transaction monitoring in Encompass: Reliable distributed transaction processing. In *Proceedings of the Very Large Database Conference*, pages 155–165, September 1981.

[8] Stefan Brand. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica, 1993.

[9] Julius Cæsar. *Cæsar's Gallic Wars*. Scott, Foresman and Company, 1935.

[10] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[11] U. S. Internal Revenue Code. Internal revenue code volume 1, 1993.

[12] U. S. Legal Code. 1989 Amendments to the Omnibus Crime Control and Safe Street Act of 1968, Public Law 101-162. United States Legal Code, U. S. Government Printing Office, 1989.

[13] Cylink Corp. CY512i press release, February 1995.

[14] Helen Custer. *Inside Windows NT*. Microsoft Press, Redmond, WA, 1993.

[15] C. J. Date. *An Introduction to Database Systems Volume 2*. Addison-Wesley, Reading, MA, 1983.

[16] Jeffrey L. Eppinger, Lily B. Mummert, and Alfred Z. Spector. *Camelot and Avalon: A Distributed Transaction Facility*. Morgan Kaufmann, 1991.

[17] Merrick Furst. Personal communications.

[18] Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, May 1987.

[19] David Golub, Randall Dean, Alessandro Forin, and Richard Rashid. Unix as an application program. In *Proceedings of the Summer 1990 USENIX Conference*, pages 87–95, June 1990.

[20] James N. Gray. A transaction model. Technical Report RJ2895, IBM Research Laboratory, San Jose, California, August 1980.

[21] James N. Gray. The transaction concept: Virtues and limitations. In *Proceedings of the Very Large Database Conference*, pages 144–154, September 1981.

[22] Louis Claude Guillou, Michel Ugon, and Jean-Jacques Quisquater. The smart card: A standardized security device dedicated to public cryptology. In Gustavus J Simmons, editor, *Contemporary cryptology: The science of information integrity*. IEEE Press, Piscataway, NJ, 1992.

[23] Nevin Heintze, J. D. Tygar, and Bennet S. Yee. Cryptographic postage indicia, 1995. To appear.

[24] Dan Hildebrand. An architectural overview of QNX. In *Proceedings of the USENIX Workshop of Micro-Kernels and Other Kernel Architectures*, April 1992.

[25] IBM Corporation. *Common Cryptographic Architecture: Cryptographic Application Programming Interface Reference*, SC40-1675-1 edition.

[26] Stuart Itkin and Josephine Martell. A PDF417 primer: A guide to understanding second generation bar codes and portable data files. Technical Report Monograph 8, Symbol Technologies, April 1992.

[27] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. Technical Report TR-31-81, Aiken Laboratory, Harvard University, December 1981.

[28] Stephen Thomas Kent. *Protecting Externally Supplied Software in Small Computers*. PhD thesis, Massachusetts Institute of Technology, September 1980.

[29] Samuel J. Leffler, Marshall K. McKusick, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Addison-Wesley, 1989.

[30] Steven Low, Nicholas F. Maxemchuk, and Sanjoy Paul. Anonymous credit cards. Technical report, AT&T Bell Laboratories, 1993. Submitted to *IEEE Symposium on Security and Privacy*, 1993.

[31] J. McCrindle. *Smart Cards*. Springer Verlag, 1990.

[32] R. Merkle. A software one-way function. Technical report, Xerox PARC, March 1990.

[33] Ryoichi Mori and Maraji Kawahara. Superdistribution: An overview and the current status. *Technical Reports of the Institute of Electronics, Information, and Communication Engineers*, 89(44), 89.

[34] National Semiconductor, Inc. iPower chip technology press release, February 1994.

[35] Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 514–523, May 1990.

[36] José Pastor. CRYPTOPOST: A universal information based franking system for automated mail processing. *USPS Advanced Technology Conference Proceedings*, 1990.

[37] Theo Pavlidis, Jerome Swartz, and Ynjiun P. Wang. Fundamentals of bar code information theory. *Computer*, 23(4):74–86, April 1990.

[38] Theo Pavlidis, Jerome Swartz, and Ynjiun P. Wang. Information encoding with two-dimensional bar codes. *Computer*, 24(6):18–28, June 1992.

[39] R. Rivest and S. Dusse. The MD5 message-digest algorithm. Manuscript, July 1991.

[40] U. S. Postal Service. Annual report of the postmaster general, fiscal year 1991.

[41] U. S. Postal Service and U. K. Royal Mail. Personal communications.

[42] Marvin Sirbu and Doug Tygar. Netbill: An internet commerce system optimized for networked delivered services. *IEEE Compcon '95 Conference*, pages 20–25, March 1995.

[43] Sean Smith, David Johnson, and J.D. Tygar. Completely asynchronous optimistic rollback recover with minimal rollbacks. In *IEEE 25th Symposium Fault Tolerant Computing*, Pasadena, CA, June 1995. To appear.

[44] Sean Smith and J. D. Tygar. Security and privacy for partial order time. In *ISCA International Conference on Parallel and Distributed Computing Systems*, pages 70–79, Las Vegas, NV, October 1994.

[45] Richard Stallman. *Gnu-emacs Manual*.

[46] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Conference Proceedings*, pages 191–200, Winter 1988.

[47] Telequip, Inc. Crypta Plus press release, January 1995.

[48] J. D. Tygar and B. S. Yee. Strongbox. In Jeffrey L. Eppinger, Lily B. Mummert, and Alfred Z. Spector, editors, *Camelot and Avalon: A Distributed Transaction Facility*. Morgan Kaufmann, 1991.

[49] J. D. Tygar and Bennet S. Yee. Strongbox: A system for self securing programs. In Richard F. Rashid, editor, *CMU Computer Science: 25th Anniversary Commemorative*. Addison-Wesley, 1991.

[50] U. S. Department of Defense, Computer Security Center. Trusted computer system evaluation criteria, December 1985.

[51] U. S. National Institute of Standards and Technology. Capstone chip technology press release, April 1993.

[52] U. S. National Institute of Standards and Technology. Clipper chip technology press release, April 1993.

[53] U. S. National Institute of Standards and Technology. Federal information processing standards publication 140-1: Security requirements for cryptographic modules, January 1994.

[54] U. S. National Institute of Standards and Technology. Csl newsletter, February 1995.

[55] Steve H. Weingart. Physical security for the μABYSS system. In *Proceedings of the IEEE Computer Society Conference on Security and Privacy*, pages 52–58, 1987.

[56] Steve R. White and Liam Comerford. ABYSS: A trusted architecture for software protection. In *Proceedings of the IEEE Computer Society Conference on Security and Privacy*, pages 38–51, 1987.

[57] Steve R. White, Steve H. Weingart, William C. Arnold, and Elaine R. Palmer. Introduction to the Citadel architecture: Security in physically exposed environments. Technical Report RC16672, Distributed security systems group, IBM Thomas J. Watson Research Center, March 1991. Version 1.3.

[58] Jeannette Wing, Maurice Herlihy, Stewart Clamen, David Detlefs, Karen Kietzke, Richard Lerner, and Su-Yuen Ling. The Avalon language: A tutorial introduction. In Jeffery L. Eppinger, Lily B. Mummert, and Alfred Z. Spector, editors, *Camelot and Avalon: A Distributed Transaction Facility*. Morgan Kaufmann, 1991.

[59] W. A. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, and F. Pollack. Hydra: The kernel of a multiprocessor operating system. *Communications of the ACM*, 17(6):337–345, June 1974.

[60] Bennet S. Yee. *Using Secure Coprocessors*. PhD thesis, Carnegie Mellon University, 1994.

[61] Michael Wayne Young, Avadis Tevanian, Jr., Richard F. Rashid, David B. Golub, Jeffrey Eppinger, Jonathan Chew, William Bolosky, David L. Black, and Robert V. Baron. The duality of memory and communication in the implementation of a multiprocessor operating system. In *Proceedings of the 11th Symposium on Operating System Principles*, pages 63–76. ACM, November 1987.

The DigiBox: A Self-Protecting Container for Information Commerce

Olin Sibert
David Bernstein
David Van Wie

Electronic Publishing Resources, Inc.
460 Oakmead Parkway
Sunnyvale, California
+ 1 408 774 6100
info@epr.com

Abstract

Information Commerce is a business activity carried out among several parties in which information carries value and is treated as a product. The information may be content, it may be returned usage and marketing data, and it may be representative of financial transactions.

In each of these cases the information is valuable and must be kept secure and private. Traditional approaches secure the transmission of that information from one point to another; there are no persistent protections. Protection of all of these components of information commerce for all parties in a transaction value chain is necessary for a robust electronic infrastructure.

A prerequisite to such an environment is a cryptographically protected container for packaging information and controls that enforce information rights. This paper describes such a container, called the DigiBox™. EPR has submitted initial specifications for the DigiBox container to the ANSI IISP Electronic Publishing Task Force (EPUB) within the User/Content Provider Standards Working Group (WG4).

1 Introduction

As services and products in modern commerce increasingly take electronic form, traditional commerce is evolving into electronic commerce. This includes both creation and enforcement of various agreements between parties in an electronic commercial relationship. It also includes enforcing the rights of these parties with respect to the secure management of electronic content or services usage, billing, payment, and related activities.

To save money, to be competitive, and to be efficient [1,2], members of modern society will shortly be using new information technology tools that truly support electronic commerce. These tools provide for the flow of products and services through creators', providers', and users' hands. They enable the creation, negotiation, and enforcement of electronic agreements, including the evolution of controls that manage both the use and consequences of use of electronic content or services. In addition, these tools support "evolving" agreements that progressively reflect the requirements of further participants in a commercial model.

Participants in electronic commerce [3,4] will need rules and mechanisms such that:

1. Information providers can be assured that their content is used only in authorized ways;

2. Privacy rights of users of content are preserved; and

3. Diverse business models related to content can be electronically implemented.

The Internet and other information commerce infrastructures will require a management component that enforces such rules, ensuring a safe, coherent, fair, and productive community. This management component will be critical to the electronic highway's acceptance. Without rules to protect the rights of content providers and other electronic community members, the electronic highway will comprise nothing more than a collection of limited, disconnected applications.

Analysts have concluded that content will constitute the largest revenue-generating component of the information superhighway [5]. It is also clear that unfettered access to content requires that content providers be able to maintain control over literary or copyrighted assets. Many analysts conclude that this will be one of the key bottlenecks in the implementation and deployment of New Media.

2 Information Commerce and Digital Value Chains

Information commerce is often considered a wholly new concept, made possible only through the use of networks and computers. In fact, a robust information economy has existed for centuries, involving trafficking in physical *representations* of information such as books, newspapers, and so on. Because such commerce involves physical goods, there is a non-negligible floor to the cost of handling information goods. The new aspects of the electronic information economy are that the information itself is the entire product and that the product can be distributed at negligible marginal cost.

The traditional information economy in physical goods is publisher-centric, because creation of information goods—particularly low-cost goods—requires a substantial manufacturing investment. Figure 1 illustrates a simplified traditional information economy: physical goods flow from a publisher (manufacturer) to a customer, in response to orders and followed by payments. The author's relationship with the publisher may be more lightweight, but the author is nonetheless dependent on the publisher to report sales and make royalty payments in accordance with the author's contract. In addition, a financial institution provides payment processing and clearing services for all parties.

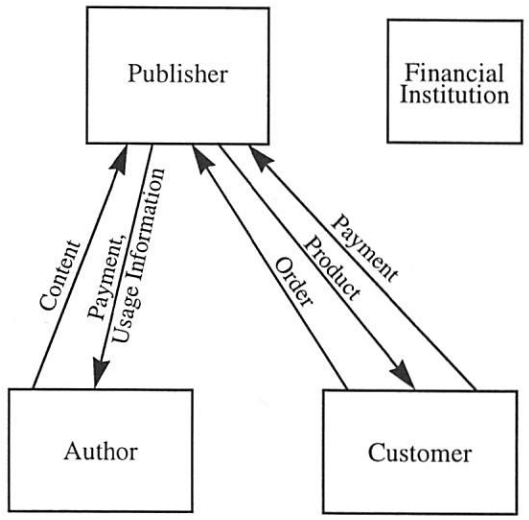


Figure 1. Traditional information economy.

Because of the flexibility afforded by electronic mechanisms, information commerce is evolving from indirect, advertiser-supported, mass-audience media to a new, niche-audience-oriented business model. In this system, members of the electronic community, with or without the economic support of advertising, pay providers directly for what they want to receive. Business-to-business purchasing is steadily evolving into a direct electronic ordering model.

Figure 2 illustrates the flexibility possible in new electronic information commerce models. Although there is still a role for publishers, this role no longer involves physical goods. Rather, the publisher is responsible for packaging and aggregating information goods and control information,

then making them available to customers. Similar to a manufacturing/distribution/retail chain for physical goods, the electronic model permits information retailers, and even end customers, to repackage and redistribute different aggregations of information while ensuring that the appropriate control rules are maintained. A clearinghouse ensures that usage information and payments are provided directly to authors and publishers; the payments themselves are made through traditional financial institutions. Because control rules are associated with information, a variety of payment and other business models can be associated with the same content (e.g., *purchase* versus *pay-per-use*).

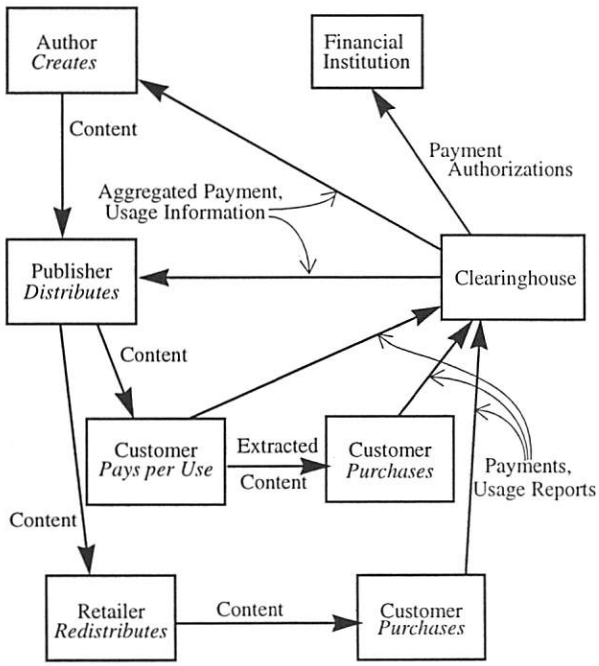


Figure 2. Electronic information economy.

The conversion from traditional commercial distribution channels requires key foundation technologies and results in a fundamental shift in existing infrastructures. This channel transformation will create a new electronic digital distribution industry. Digital distribution employing the DigiBox container architecture and its associated support environment, InterTrust™, can play a critical role in this transformation of the communication, media, and information technology markets.

2.1 Protecting All the Information in Information Commerce

The very properties that make "the net" attractive as a distribution medium—ease of manipulating information in electronic form—also appear to make these protections intractable. Addressing this dichotomy requires a paradigm shift in computer architecture to introduce the concept of a "secure processing" environment in which protected information can be manipulated without being subject to external tampering or disclosure. A prerequisite to such an environment is a cryptographically protected "container" for seamlessly packaging information and controls that enforce information use rights.

The DigiBox described by this paper is such a container.

The need for various information commerce computers and appliances to interoperate requires that this container format and its access methods be standardized. EPR has submitted initial specifications for the DigiBox container to the American National Standards Institute (ANSI) Information Infrastructure Standards Panel (IISP) through the Electronic Publishing Task Force (EPUB) in the User/Content Provider Standards Working Group (WG4).

The primary goal of information protection is to permit proprietors of digital information (i.e., the artists, writers, distributors, packagers, market researchers, etc.) to have the same type and degree of control present in the "paper world." Because digital information is intangible and easily duplicated, those rights are difficult to enforce with conventional information processing technology. Many types of rights (compensation, distribution, modification, etc.) are associated with the various elements of information commerce, and these information property rights take many forms. At a high level, there is the legal definition of "copyright," codified in U.S. law [6–9] and the Berne Convention. This gives copyright holders a legal right to control how copyrighted information is handled. In addition, various high-level rights are conferred by contractual arrangements between primary rightsholders and other parties.

For example, the protections needed for content elements incorporate the licensing provisions for the intellectual property rights of the content rightsholders. In a broader sense, these rights include control over several activities: the right to be compensated for use of the property; the right to control how content is distributed; the right to prevent modification of content by a distributor; "fair use" rights; the rights to the usage data, privacy rights of individuals, and so on.

In the realm of physical goods, these rights are enforced by a combination of legal and technical means. However, the technical means can be (and are) unsophisticated because the technology for violating rights is relatively expensive and time-consuming—in comparison to equivalent activities with respect to digital information. Photocopying a book or copying a video cassette is inherently more labor intensive and costly than copying a file. So, while defeating technical means of enforcement is (relatively) expensive, it can be done—and often the legal means to deter this are inadequate.

2.2 Information *Commerce*—Not Just Payment

Rights protection is also a fundamental aspect of commerce. Commerce is not just a way for two parties to pay each other for something. Rather, it is an extraordinarily rich web of relationships among parties that concerns payment, negotiation, control, advertising, reporting, auditing, and a variety of other activities. These activities are important aspects of the transaction relationships. Often the information carried in these reports, audits, and the like is highly valuable and highly confidential, perhaps even more valuable than the content that is the subject of the information commerce at hand. These activities too are performed and controlled in the "paper world" by legal and technical means, but there are no widely used models for their electronic equivalents.

Figure 3 shows some of the operations that could occur in true electronic commerce, using the Internet World-Wide Web [10] mechanisms as an example. Creators originate content and apply rules (e.g., "pay author $1.00/use") for its use. Distributors repackage content, applying additional rules (e.g., "pay $5.00 for the collection, then pay the creator," "report use of each item"). Users receive content and operate on it, generating billing reports and usage reports that are delivered to a clearinghouse and paid or summarized back for the originating parties. This structure is very rich and is capable of supporting many business models. There are multiple flows of information in many different directions amongst the parties involved in the transactions.

Another example is that of an advertiser (acting as distributor, or with a distributor). The advertiser might have a rule that offers a discount, or no charge at all, but only if the user views the advertisement and agrees to have that fact reported to the advertiser.

It is relatively simple to devise schemes for parties to pay each other electronically (for example, DigiCash [11], NetBill [12], Open Market [13], SNPP [14], NetCheque [15], First Virtual [16], etc.). Payment, however, constitutes only one—and perhaps the simplest one—of the means in which parties in commerce interact. All the other information commerce components must be accomplished with the same needs for security, privacy, and integrity. In fact, these aspects of electronic commerce, including rights protection, are strongly intertwined in the digital economy, because much digital commerce concerns information and innovative business models for information commerce.

3 Existing Approaches to Information Commerce

Information proprietors employ a variety of technological protection approaches today. These approaches are generally "point solutions," in that they protect a specific type of property in a specific context and enforce only specifically defined rights—typically only the right to compensation for use. Because the technologies are limited, the market is fragmented, and there are no general protection solutions.

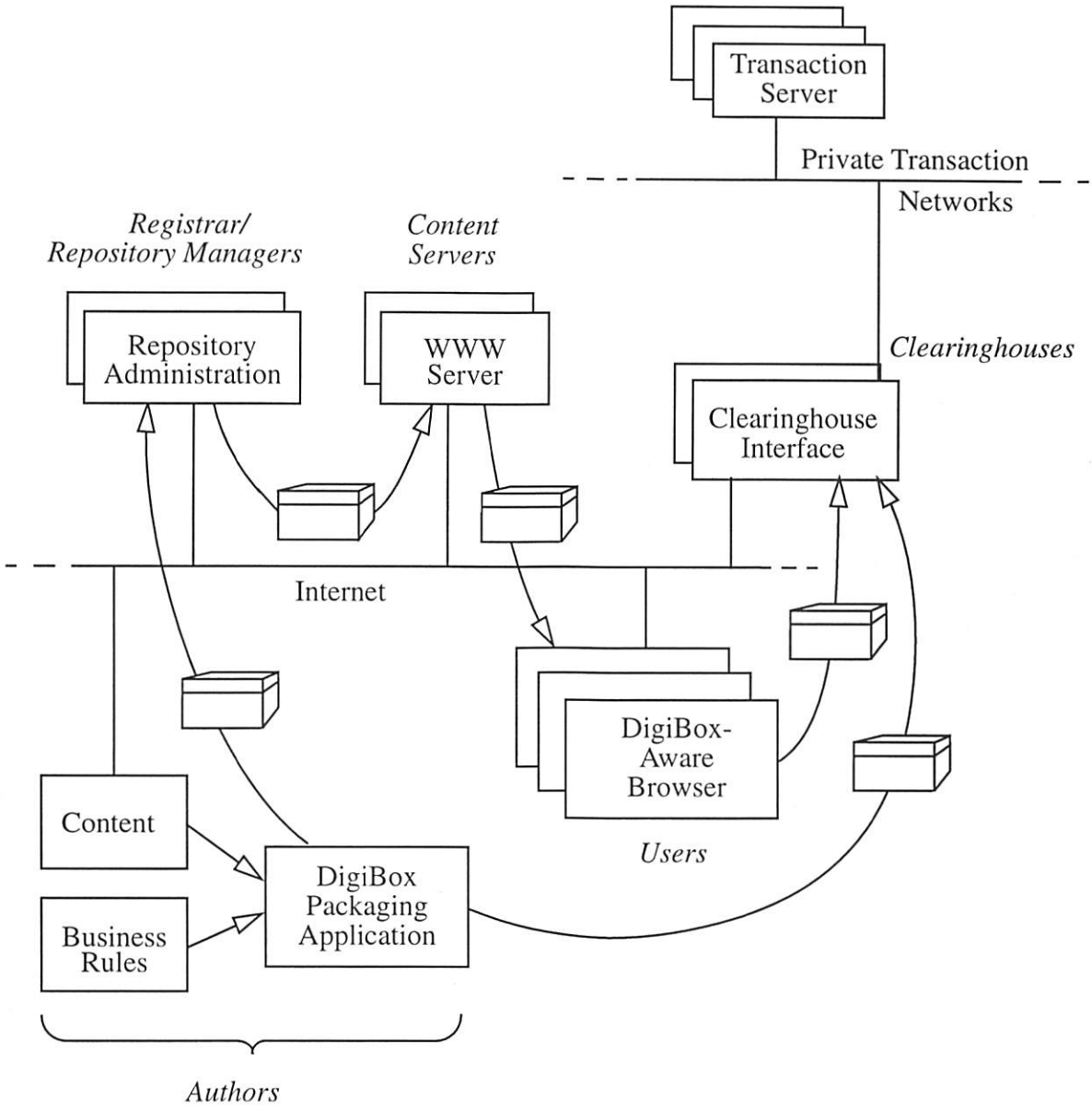


Figure 3. Multi-party Internet information commerce.

3.1 No Protection

Much digital property is distributed without any technological enforcement for property rights, on the assumption that legal means suffice. This approach works well enough for many low-value properties, but it has the disadvantage of raising the price to legitimate users who must pay for both their own and illegitimate use. In many cases, however, this cost is negligible, and no protection is an economically sound choice. Even for content that is free, however, a creator may wish to impose some rules for reporting or some access control. Of course, privacy rights of users will be a concern to many.

3.2 License Managers

For some valuable software properties, license managers are used. Because a software property is dynamic (executable), it is feasible to restrict it so that it functions properly only through interaction with a license manager process. In general, there is no protection of usage data in these schemes. In some cases this technique has been applied to content protection, but only with limited success [17, 18].

3.3 Cryptographic Unlock

Some static properties (fonts, for example; also some installable software) are protected by a simple "unlock" scheme: a purchaser makes a purchase, for example by telephone with a credit card, and receives a cryptographic key in return. This key can then be used to "unlock" one property from some widely distributed medium (e.g., CD-ROM or network download). This mechanism is relatively inflexible, and its inherently manual nature makes it expensive.

3.4 Billing Schemes

Various billing schemes (as mentioned above) permit purchase of information following what is essentially an electronic check or electronic credit draft model. These methods are suitable for conventional transactions, but not for the enormous volumes of (individually) very low-value transactions that would be generated using a complex digital property.

3.5 Secured Delivery

Various secured delivery systems (e.g., SSL [19], SHTTP [20]) share the same problems as cryptographic unlock, but in a network context. They are only point-to-point solutions, with the information (content, usage data, etc.) at each site being left unprotected once the delivery has occurred. Furthermore, they are inherently online systems: it is not practical to decouple the delivery of information from payment for its use.

4 Information Protection Architecture: InterTrust and DigiBox

EPR has produced the InterTrust Virtual Distribution Architecture to solve unmet, critical needs of electronic commerce. Almost any imaginable information transaction can be supported by InterTrust. A few examples include distribution of content (e.g., text, video, audio) over networks, selective release of data from a database, controlled release of sensitive information, and so on. InterTrust can also support the secure communication of private information such as EDI and electronic financial transactions, as well as delivery of the "back channel" marketing and usage data resulting from transactions.

DigiBox is a foundation technology within InterTrust. It provides a secure container to package information so that the information cannot be used except as provided by the rules and controls associated with the content. InterTrust rules and controls specify what types of content usage are permitted, as well as the consequences of usage such as reporting and payment.

Within InterTrust, DigiBox containers can enforce a "distributed electronic contract" for value-chain activities functioning within an electronic distribution environment. This unique approach underlies EPR's information metering and digital rights protection technology. Electronic commerce infrastructure participants can use InterTrust to substantially enhance their network, security, or payment method solutions.

The DigiBox is a container for both digital property (content) and controls. It is used in conjunction with a locally secured rights protection application (discussed further below) to make content available as governed by arbitrarily flexible controls.

The DigiBox container mechanism is implemented in a set of platform-independent class libraries that provide access to objects in the container and extensions to OpenDoc and OLE object technologies. DigiBox allows rights management components to be integrated with content in highly flexible and configurable control structures. Digi-

Box rights management components can be integrated with content in a single deliverable, or some or all of the components can be delivered independently. DigiBox rights management components enable true superdistribution [21] and can support virtually any network topology and any number of participants, including distributors, redistributors, information retailers, corporate content users, and consumers.

4.1 Content

The digital information in a DigiBox (one or more "properties") is information in any form. It may be mapped to a specific compound object format (e.g., OpenDoc, OLE, PDF), or may be application specific.

Further, it may be delivered in stream or other communication-oriented forms, not just in a file-like container.

4.2 Controls

Controls specify rules and consequences for operations on content. Controls are also delivered in a DigiBox, and the controls for a property may be delivered either with the property or independently. Controls are tied to properties by cryptographic means.

Because controls can be delivered with properties in a container, the DigiBox supports superdistribution.

4.3 Commerce

Commerce takes place governed by controls. This may involve metering, billing for use, reporting of usage, and so on. These operations take place locally in a secure environment, and they generate audit trails and reports that must be reported periodically to clearinghouses.

5 DigiBox Implementation

The DigiBox is a structure that can hold, in a protected manner, information commerce elements of all kinds: content, usage information, representation of financial transactions (e.g., electronic cash), and other digital elements of information commerce.

5.1 Container Logical Structure

Figure 4 shows the logical structure of properties and control sets in two containers. Container C_1 holds two properties, P_1 and P_2, and one control set, CS_1, that applies to property P_1; container C_2 contains two control sets and no properties. As shown in the example, each of these elements has a title attribute to provide a human-readable description of the element and, for control sets, an attribute indicating to what other elements the control set applies.

A control set specifies rules and consequences, such as pricing, reporting, and so on, for the properties to which it applies. A user holding just this container could use (e.g., view, print) content from P_1—though only as specified by CS_1. Because there is no control set applying to P_2 in that container, P_2 would not be usable in any way.

A user holding both containers could use property P_2, as specified by CS_2, and in addition has the choice of whether to designate CS_1 or CS_3 when using P_1. CS_3, which describes itself as "discount," is likely to be the user's preferred choice.

The DigiBox includes several elements: organizational structures, properties, controls, and supporting data items. Almost all the information in a DigiBox is encrypted, as described below, and access to the encrypted form is provided through a storage manager as appropriate, depending on how the DigiBox is delivered (e.g., as a file or as a data stream).

5.2 Container Physical Structure

Figure 5 is a schematic picture illustrating the physical structure of a DigiBox container. (Some elements have been omitted for clarity.) It begins with a *container header* structure containing descriptive and organizational information about the container. Part of the container header is encrypted (both for secrecy and for integrity protection); the rest is public organizational informa-

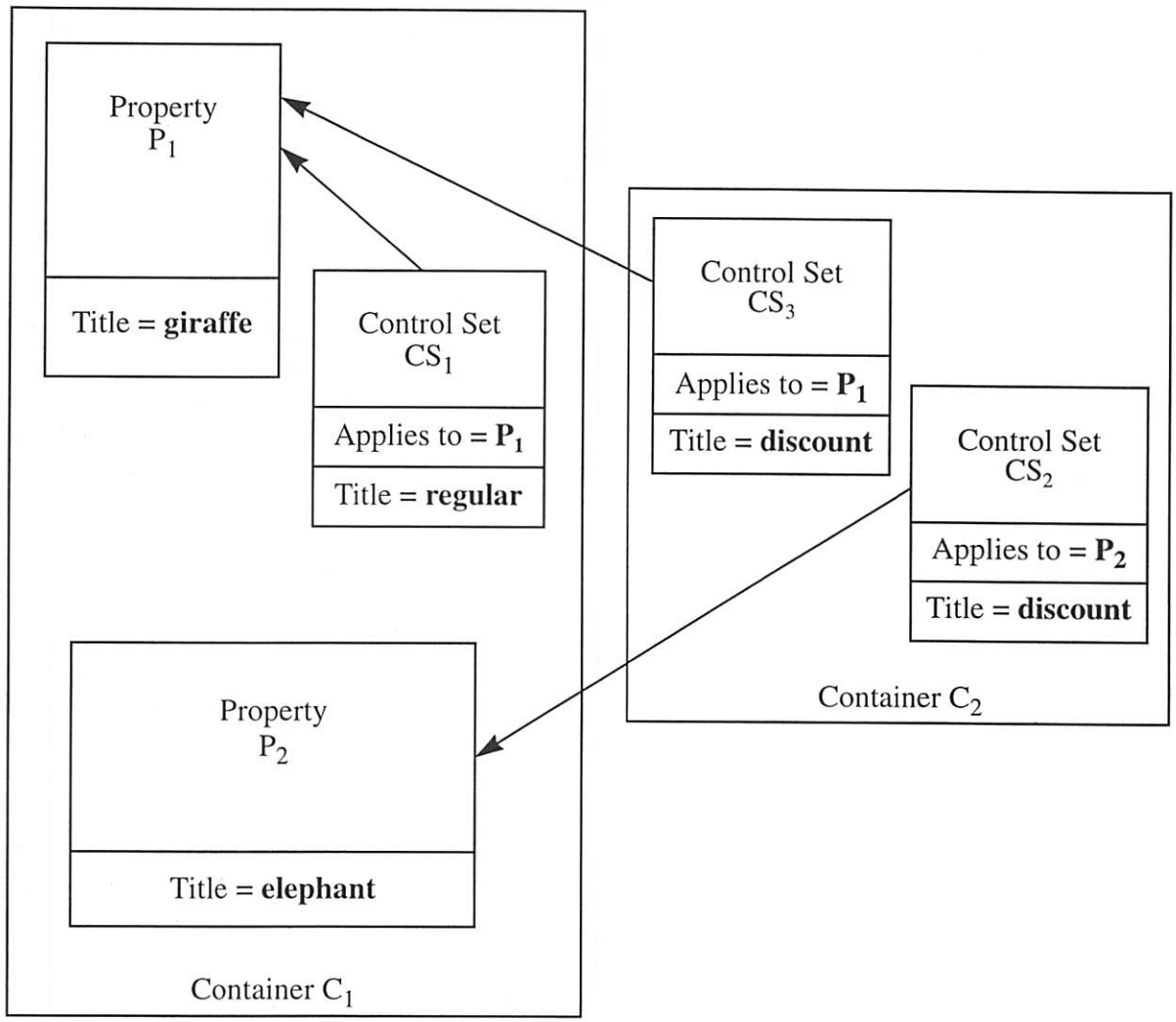


Figure 4. Container logical structure.

tion. The header is followed by additional container-wide structures such as the *transport key block (TKB)* and the *container table of contents (TOC)*, some of which are encrypted and others not.

These organizational elements are followed by the structures defining the container's content (e.g., *properties* and *control sets*). As shown in the figure, a property is represented by a *property header, property attributes,* and data blocks composing the property. As shown, the header is encrypted and the attributes are not; the data blocks may be wholly or partly encrypted, or not at all, depending on security requirements.

The figure shows an example property consisting of a multimedia property formed from a pair of synchronized data streams for audio and video. In this example, each video block is mostly unencrypted so that access can be rapid while still maintaining reasonable security—encrypting even 10 percent of an MPEG stream renders it effectively useless for illicit copying. On the other hand, the audio is entirely encrypted, and each audio block

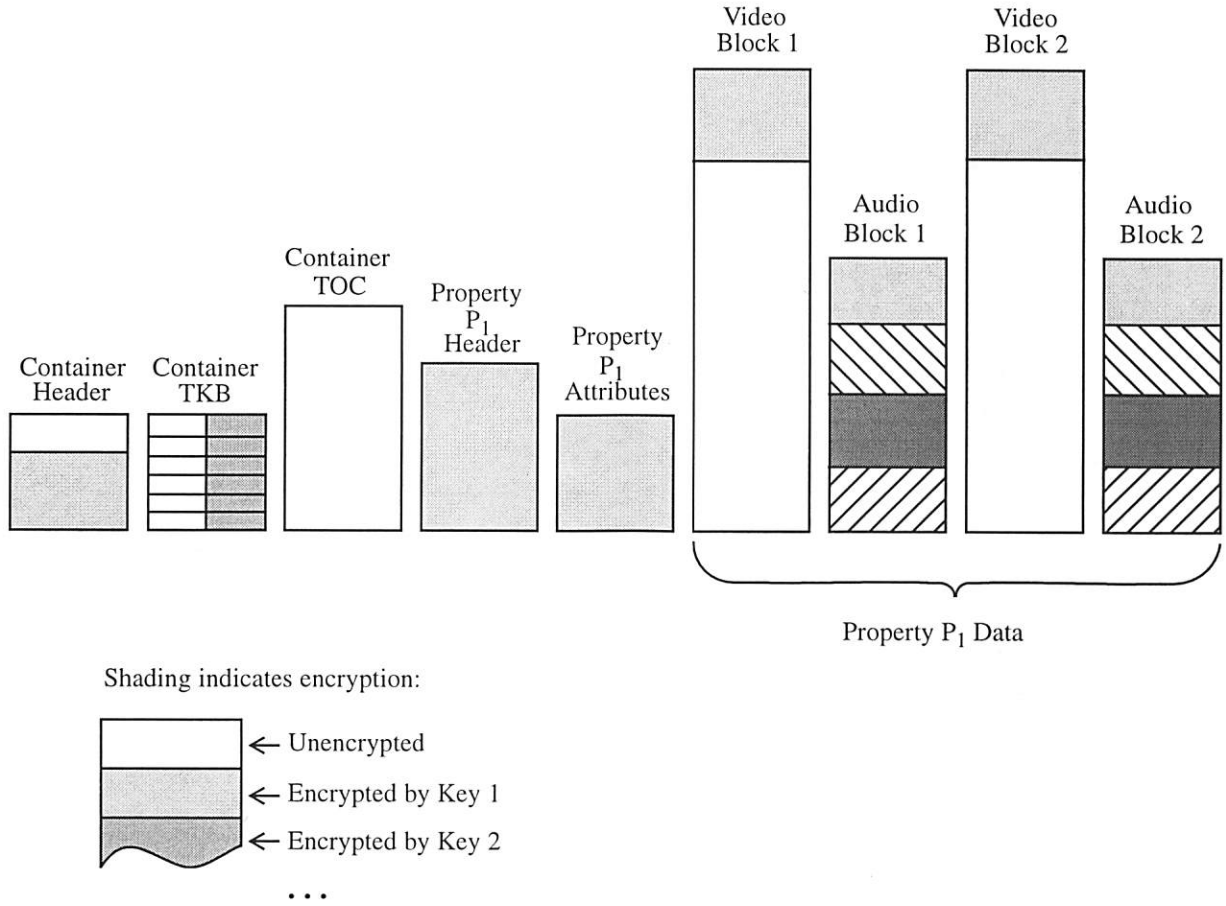


Figure 5. Container physical format.

uses four distinct keys, because the content proprietor requires much stronger security for audio than for video.

A property is represented as one or more property sections, each of which is independently associated with control information, and which may also be stored and accessed independently. A property, for example, might be a collection of clip-art images, and each image might be a property "chunk," with its own control specifying how that image's creator is compensated.

Controls can map to property chunks at arbitrary granularity and can enforce arbitrary organizational structures within the property (such as a file hierarchy). Controls can apply to individual bytes, frames of a movie, segments of a musical piece, and so on, because the mapping is performed by a control process specified by the control structure, not simply via a table-driven data structure.

5.3 Cryptographic Techniques

The high-level elements in a DigiBox are encrypted with a *transport key* that is normally derived (by exclusive OR) from two parts: one that is delivered in the DigiBox itself, encrypted with a public key algorithm, and the other that is stored in protected storage locally. The locally stored part is shared among all the local nodes capable of processing that DigiBox, but the part in the DigiBox is unique. This separation provides protection against accidental or malicious disclosure of either part.

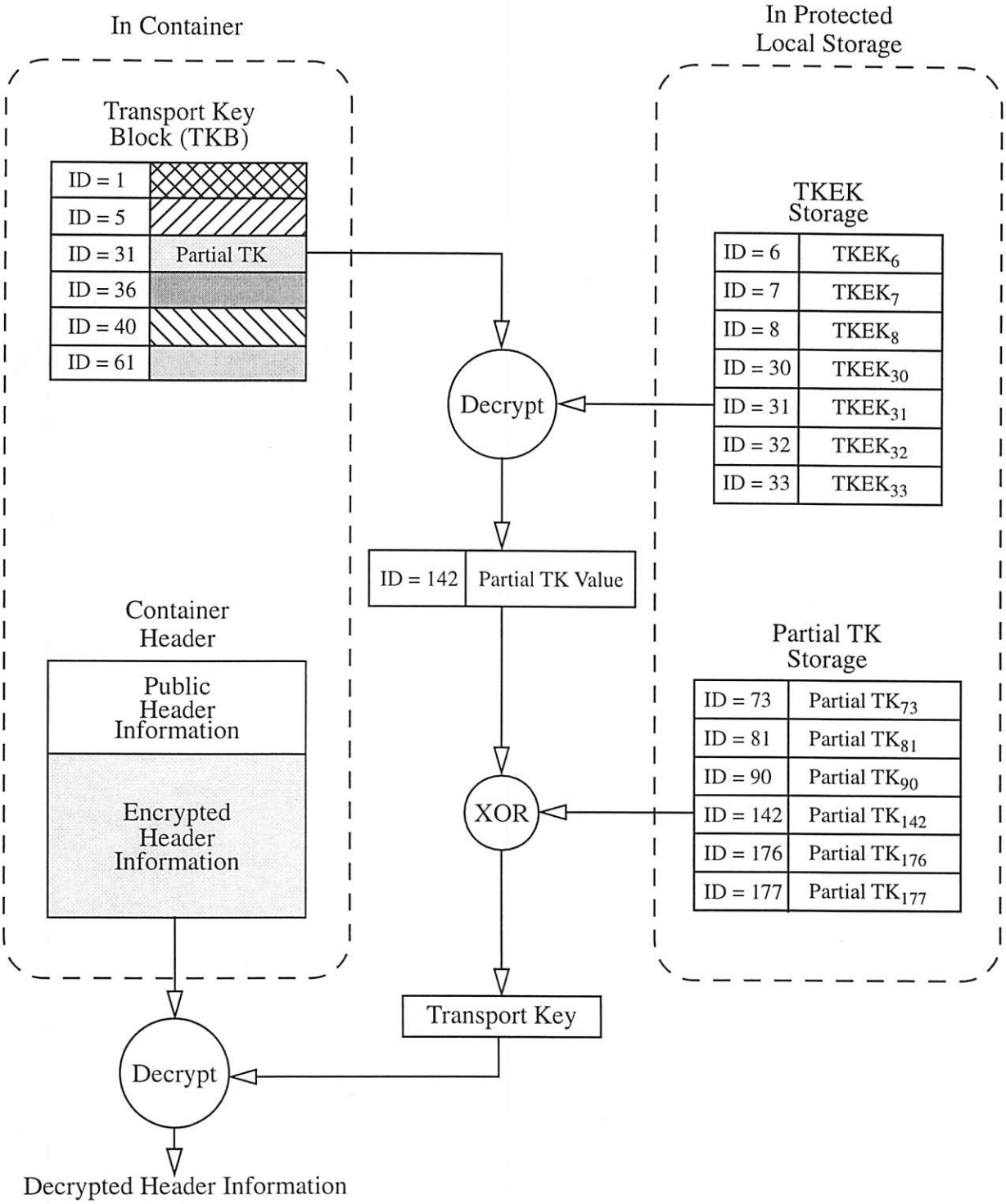


Figure 6. Container transport security.

Figure 6 illustrates how the transport key (TK) is derived. The transport key block (TKB) contains one or more slots, each of which contains a partial transport key encrypted under a different transport key encrypting key (TKEK). Each TKB slot identifies the TKEK used, and a matching TKEK is

selected from local protected storage. Decrypting the slot yields a partial TK, which is combined with its corresponding partial TK again from protected local storage to yield the actual TK for decrypting the container header.

The data for the property itself is encrypted with other keys ("content keys") that are themselves delivered in encrypted high-level structures; this approach permits the keys for a property to be delivered entirely separately from the property or its controls. Multiple keys, in a wide variety of key-mapping schemes, are used to encrypt the data, limiting the loss that would occur from disclosure of any one key.

All DigiBox control structures are both encrypted and verified for integrity with a cryptographic hash function. Several cryptographic algorithms are supported for these control structures (principally for export control reasons), and arbitrary algorithms are supported for encryption of the data.

5.4 Security Characteristics

The DigiBox cryptographic structures are designed to be secure even in the face of loss of individual key components, and to minimize the damage in case a key or processing environment is compromised. The system is designed to provide commercially acceptable risks and losses for a variety of business models.

The basic algorithms are strong: Triple DES [22] and RSA [23] are preferred. This security is, of course, only as strong as the tamper-resistance of the local processing environment. The preferred implementation of DigiBox processing relies on a "secure processing unit" (SPU) that contains a CPU, memory, program storage, and key storage in a single tamper-resistant hardware package. Although these are not widely available today, the variety of applications they might support makes it likely that such SPUs will become widely integrated into common computing platforms. When running in an SPU, the DigiBox processing and control mechanisms are sufficiently well protected to support most commerce applications.

In the absence of an SPU, other approaches are useful for many business models. In fact, a software-only implementation is sufficient for many applications, because much content is of relatively low value and is used in a context (business to business) where a modest level of fraud is both less likely and more tolerable. As long as the software is moderately difficult to defeat and tools to defeat it have no legitimate purpose, business models can be supported where some risk of loss is acceptable. In the world of electronic commerce, just as for traditional commerce, security is not absolute: it is just a factor to balance against the cost of loss and fraud.

6 Conclusions

The DigiBox is one component of a general-purpose electronic commerce solution that rests on three basic principles: rights protection, interoperability, and strong security.

Electronic commerce, and information commerce in particular, needs a robust information protection mechanism, including rights protection and controls, not just payment systems. As the electronic world evolves, however, and moves forward from simply emulating traditional transactions into entirely new business models, rights protection and control will become the predominant issues.

Protection of intellectual property rights in information requires strong cryptography as well as a flexible infrastructure for controlling use of the information. A standard protected container for information is necessary to support interoperability—most existing schemes tightly bind the creator of protected information and the software that processes it. A standard container can rationalize information commerce and reduce costs for all participants.

In the long term, general-purpose secure electronic commerce will need pervasive deployment of tamper-resistant hardware devices to perform secure processing of protected content. However, as these solutions are developed, many business models can be accommodated with weaker or less complete solutions because the risk and expected losses are commercially acceptable.

Business-to-business purchasing is steadily evolving into a direct electronic ordering model. Future communications and media markets will become increasingly segmented and specialized in response to customer preferences and needs and involve increasing, and more sophisticated, direct interaction between consumers and providers. These markets and their value chains (with or without intermediary distributors) will require secure metering and control tools that enable a user to efficiently and economically tailor resources to his or her own desires.

During the next decade, digital delivery of traditional electronic products, such as information databases and software, will be joined by a rapidly growing array of both New Media and electronically distributed traditional content. The conversion from traditional models requires key foundation technologies and will result in a fundamental shift in current infrastructure. This transformation will create a new distribution industry. Digital distribution employing a universal content and commerce container can play a critical role in this broad economic transformation.

7 References

[1] A. Chandler and H. Daems, "Administrative Coordination, Allocation, and Monitoring: A Comparative Analysis of Accounting and Organization in the U.S.A. and Europe," *Accounting, Organizations and Society,* 1979: 3–20.

[2] O. Williamson, "The Modern Corporation: Origin, Evolution, Attributes," *Journal of Economic Literature* XIX (1981): 1537–1568.

[3] Office of Technology Assessment, *Accessibility and Integrity of Networked Information Collections.* Washington, D.C.: U.S. Government Printing Office, July, 1993.

[4] E. Hollings, *Communications Competitiveness and Infrastructure Modernization Act of 1990.* Washington, D.C.: U.S. Government Printing Office, report of the Senate Committee on Commerce, Science, and Transportation, 12 September 1990.

[5] R. Benjamin and R Wigand, "Electronic Markets and Virtual Value Chains on the Information Superhighway," *Sloan Management Review,* Vol. 36 No. 2 (1995).

[6] U.S. Constitution, Article 1, Section 8, Clause 8 (1787).

[7] U.S. Copyright Act of 1978

[8] 17 U.S.C. s107

[9] 17 U.S.C s102(a)

[10] T. Berners-Lee, R Caillian, and J.-F. Groff, "The World Wide Web," *Computer Networks and ISDN Systems,* Vol. 25 (Dec. 1992), pp 454–459.

[11] D. Chaum, "Achieving Electronic Privacy," *Scientific American,* August 1992, pp 96–101.

[12] M. Sirbu and J. D. Tygar, "NetBill: An Internet Commerce System," *IEEE CompCon Proceedings*, March, 1995, pp 20–25.

[13] D. Gifford et al., "Payment Switches for Open Networks," *IEEE CompCon Proceedings,* March, 1995, pp 26–31.

[14] S. Dukach, "SNPP: A Simple Network Payment Protocol," MIT Laboratory for Computer Science, Cambridge, MA, 1993.

[15] B. C. Neuman and G. Medvinsky., "Requirements for Network Payment," *IEEE CompCon Proceedings,* March, 1995, pp 32–36.

[16] First Virtual, Inc. "Introducing the First Virtual Internet Payment System," 1994.

[17] A. K. Choudhury, et al., "Copyright Protection for Electronic Publishing over Computer Networks," June 1994, *IEEE Network Magazine.*

[18] J. Erickson, "A Copyright Management System for Networked Interactive Multimedia," *Proceedings of the 1995 Dartmouth Institute for Advanced Graduate Studies,* 1995.

[19] K. Hickman, "SSL Reference Manual," Netscape Corporation World Wide Web Site, `http://www.netscape.com/newsref/std/sslref.html`, 1994.

[20] E. Rescorla and A. Schiffman, "The Secure HyperText Transfer Protocol," Internet Draft draft-resorla-shttp-0.txt, 1994.

[21] B. Cox, "Superdistribution," *Wired*, Sept. 1994, pp 89-92.

[22] U.S. National Bureau of Standards, "Data Encryption Standard," *Federal Information Processing Standards Publication,* FIPS PUB 46-1, Jan. 1988.

[23] R. Rivest, A. Shamir, and L. Adleman, "On Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, Vol. 21 (Feb. 1978), pp 120–126.

Kerberos Plus RSA for World Wide Web Security

Don Davis*

August 3, 1995

Abstract

We show how to use Kerberos to enable its clients to interact securely with non-Kerberized World Wide Web servers. That is, our protocol does not require that the Web server be a member of a Kerberos realm, and also does not rely on time-synchronization between the participants. In our protocol, the Kerberos client uses the Web server's public-key certificate to gain cryptographic credentials that conform to public-key authentication standards, and to SHTTP. The client does not perform any public-key encryptions. Further, the client is well-protected from a man-in-the-middle attack that weakens SSL. Our protocol conforms to the current specifications for the Kerberos protocol and for the Secure Hypertext Transfer Protocol.

1 Introduction

The effort to secure the World-Wide Web for electronic commerce brings to a sharp focus the difficulties that have prevented the Internet from adopting a single coherent security model. On the one hand, like the Internet, the Web as a whole is too big to bring into a centrally-administered key-distribution system like Kerberos. [5, 13, 8] Indeed, the whole point of the Web is to bring together clients and servers across great distances, so it seems that public-key security is the only natural model, and the only security model that can be made to work. On the other hand, though, it's unrealistic to suppose that all Web clients will have public-key certificates any time soon. Public-key cryptography costs more money and more cycles than many home-computer users want to spend on an invisible benefit.

Netscape's Secure Socket Layer protocol [7] attempts to finesse this problem by waiving the client's authentication to the server, so that only the server needs a public-key certificate. This approach prevents the Web server from detecting credit-card fraud, which puts all credit-card holders at risk. CommerceNet's proposed standard, S-HTTP [10], provides for full mutual authentication, and supports several varieties of public-key and private-key cryptography. However, S-HTTP cannot do anything to bring these several varieties into cooperative communion; it enables public-key clients to shop at public-key-authenticated Web pages, and brings Kerberos-equipped clients and servers together, but cannot bring the two groups together.

What Web commerce needs now is a lot of Web users who have cryptographic credentials of some sort, and some way to bring the clients' and servers' disparate kinds of credentials together. We believe that the natural way to authenticate most home-computer users is with trusted third-party key-distribution, such as Kerberos provides. Internet access providers have to do password and account administration anyway, and some of them are beginning to install Kerberos. We also believe that public-key cryptography is more natural for authenticating the Web servers. We propose as a solution a merger or marriage between these competing styles of cryptography, with Kerberos taking responsibility for bridging the differences. Our protocol combines Kerberos' performance advantages with public-key's terrific geographic reach.

Other mergers between Kerberos and RSA have been proposed. MIT's Schiller and Atkins have built a Kerberized service that certifies PGP public keys [15] for Project Athena's users [12]. Neuman et al. have preliminarily proposed in an Internet Draft that Kerberos can help its clients manage their RSA keys [9]. Finally, the present paper's protocol is derived from a protocol that we presented in [3].

2 Problem Statement

We assume that each Web user will share a password with at least one Kerberos service, which typically will be administered by the user's commercial

*Affiliations: Independent Consultant, 1318 Comm. Ave #16 Allston, MA 02134; *don@mit.edu*

Internet-access provider. Some users might get their net access and credentials via an employer or a university network. We don't assume that these Kerberos servers are linked in interrealm trust relationships. We don't assume that the Kerberos clients are capable of public-key encryption operations. Finally, we don't assume that the Kerberos clients' clocks are synchronized, because of a recent finding that Kerberos' requirement for synchronized clocks can be relaxed [1].

We assume that each commercial Web server will gain an RSA public-key certificate [11], that the server will support S-HTTP, and that the server can perform a few styles of symmetric-key encryption, including DES, 3DES, and RC4. We do not assume that the Web servers can contact a Kerberos server, and we don't assume that the Web servers have any administrative relationship with a Kerberos service.

With these assumptions, we want to extend Kerberos and S-HTTP as little as possible, so as to enable any Kerberos-authenticated client to mutually authenticate with any public-key-bearing Web server. Thus, the client and server need to share a symmetric session-key. It turns out to be surprisingly easy to issue such a session-key.

3 Protocol Notation and Review

To clarify our cryptographic terms and notation, this section will quickly review the Kerberos protocol. [1] Because S-HTTP supports Kerberos as an authentication option, we don't refer to S-HTTP details, and don't need to review the S-HTTP protocol itself.

Before proceeding, we must clear up a jargon collision: the Kerberos and RSA communities use the term "private key" in different ways. To replace "private key," we will use "secret inverse key" for the non-public RSA keys, and we'll use "password key" for a Kerberos client's identifying DES key. The other types of key we'll talk about are RSA public keys and symmetric session keys. These last may be RC4, IDEA, DES, or other keys. We'll denote public keys by P_a, secret inverses by P_a^{-1}, password keys by K_a, and session keys by $K_{a,b}$.

Now, suppose Alice is a client of a Kerberos server KRB, so that she shares a password key K_a with the server. When Alice logs in, KRB can use her password key to securely issue a session-key for her to use later:

$$A \rightarrow KRB \quad : \quad A,\ KRB \tag{1}$$

$$KRB \rightarrow A \quad : \quad \{KRB,\ L,\ K_{a,krb}\}^{K_a},\ \{A,\ L,\ K_{a,krb}\}^{K_{krb}} \tag{2}$$

Alice's request says that she wants a ticket for the Kerberos service, The Kerberos server replies with two similar encrypted messages, which identify Alice and KRB to each other as the only bearers of a new symmetric session-key $K_{a,krb}$. Each message also limits the key's lifetime to a fixed span L. The second message is called Alice's Ticket-Granting Ticket. We denote it $T_{a,krb}$, or TGT informally.

Alice has not yet proven her identity to anyone; her request was entirely in plaintext. As we will see below, her TGT enables her to share session-keys with other services, and she'll use those keys to authenticate herself to them.

4 Our Proposed Solution

Our approach to Web security is to equip each Kerberos server with the ability to RSA-encrypt its key-certifying tickets. For this to work, each Kerberos server will need a public-key pair P_{krb}, P_{krb}^{-1} of its own. To prepare a symmetric session-key for a Web server, a KDC will encrypt with its own secret-inverse key, and again with the Web server's public key.

So, suppose Alice contacts Bob's commercial Web server; when she decides to order something from Bob, she requests his public-key certificate:

$$A \rightarrow B \quad : \quad \text{"cert. request"} \tag{3}$$

$$B \rightarrow A \quad : \quad C_b,\ C_{ca} \tag{4}$$

Here, C_b is Bob's public-key certificate $\{B,\ P_b\}^{P_{ca}^{-1}}$, which asserts that P_b is Bob's public key, and which is signed with the secret-inverse key P_{ca}^{-1} of Bob's Certification Authority CA. Similarly, C_{ca} is CA's certificate; in practice, Bob may send Alice a chain of CA certificates, and not just one [4].

Now, Alice sends Bob's credentials and her own to her Kerberos server KRB, in a request for a new session key and ticket:

$$A \rightarrow KRB \quad : \quad B,\ C_b,\ C_{ca},\ T_{a,krb},\{time\}^{K_{a,krb}} \tag{5}$$

$$KRB \rightarrow A \quad : \quad \{B,\ L,\ K_{a,b}\}^{K_{a,krb}},\ \{\{A,\ L,\ K_{a,b}\}^{P_b}\}^{P_{krb}^{-1}}, C_{krb} \tag{6}$$

Alice's request is essentially a standard user-to-user ticket request [2, 8], except that instead of sending a TGT $T_{b,krb}$ for Bob, she sends his certificate-chain.

[1] For clarity, we avoid mentioning the Ticket-Granting Service [8] by name, and we omit the details of how to avoid synchronizing clocks [1].

The last part of her request is an encrypted timestamp. To service Alice's request, the Kerberos server first validates Bob's public key P_b with his Certification Authority's key P_{ca}. Kerberos' response is also standard, except that Alice's ticket is RSA-encrypted twice, instead of being DES-encrypted as is usual. [2] Kerberos uses P_b to encrypt the new session key $K_{a,b}$, and finally uses its own RSA secret-inverse key P_{krb}^{-1} to sign the encrypted key.

Alice uses her daily Kerberos session-key $K_{a,krb}$ to decrypt the first part of Kerberos' reply, gaining her new session-key $K_{a,b}$. This enables her to mutually authenticate with Bob, and to send her credit-card number N_{cc} to him securely:

$$A \rightarrow B \quad : \quad \{\{A,\ L,\ K_{a,b}\}^{P_b}\}^{P_{krb}^{-1}}, C_{krb}, \{A,\ N_{cc},\ time'\}^{K_{a,b}} \qquad (7)$$

$$B \rightarrow A \quad : \quad \{B,\ time' + 1\}^{K_{a,b}} \qquad (8)$$

When Bob receives the doubly-encrypted session key from Alice, Kerberos' outer signature assures him that it was Kerberos who performed the inner encryption, so Bob believes that only Alice has seen the key $K_{a,b}$. Bob then uses $K_{a,b}$ to decrypt Alice's timestamped charge-number, and returns the timestamp to her.

Bob trusts Alice's Kerberos server KRB to identify her truthfully. For this arrangement to work, each Kerberos server's certificate-chain will have to include a signed authorization certificate, identifying the server as a legitimate issuer of session-keys. Bob checks this certificate for authenticity when he checks the rest of KRB's certificate-chain.

The Kerberos service KRB does not access the key database when it answers Alice's request for Web tickets. Thus, KRB can actually be an ancillary, diskless key-server, which would be located close to Alice. For example, her online-access provider would probably put such a Web key-service in each of its local dialin offices. This gives Alice better response-time, and relieves her central Kerberos server of a substantial work-load. Alice would get her ticket $T_{a,krb}$ and session-key $K_{a,krb}$ from Kerberos in the usual way.

5 Discussion

By relieving clients of the need for public-key certificates, this protocol lowers users' entry-level costs for Web commerce. Since most home customers will probably use their online-access providers' browsers, this really means that the OA providers won't have to pass per-client fees for RSA certificates and licensing on to their customers.

Our protocol concentrates all RSA encryption operations at the servers. The Web-ticket servers are easily replicated, so this concentration does not create a bandwidth bottleneck. This means that Alice's CPU waits idly for her server to perform these slow encryptions. She sees about the same real-time delay as she would have if she had done the RSA operations herself, but her CPU is free for other tasks, such as rendering. This makes better use of everyone's computational resources, especially if the servers run with RSA encryption hardware.

Another advantage of our proposal is that it is the Kerberos server who verifies Bob's certificate with the top-level CA's public key. This is better than having Alice check the certificate, because Alice receives her copy of P_{ca} in her browser's executable. With casual freeware distribution of browsers, Alice is unlikely to know whether her browser's P_{ca} has been altered by an attacker. If it has, Alice is vulnerable to a "man-in-the-middle" attack that collects credit-card numbers. (When Alice requests Bob's certificate, the attacker intercepts it, and uses it to set up a secure connection with Bob, in Alice's name. Meanwhile, the attacker sends Alice a forged certificate C_b for Bob, signed with the fraudulent CA secret-inverse key P_{ca}^{-1}. Thus, when Alice sets up her connection, she believes that she's corresponding securely with Bob. The attacker forwards all of Alice's and Bob's messages faithfully, but records Alice's credit-card number for later use.) We suggest that it is natural for users to rely on a trusted third-party like Kerberos to manage the top-level CA key, so as to block this attack. Similarly, the Kerberos server (or the Web-ticket server) can manage the public-key Certificate Revocation List for Alice.

Our proposal does present a minor difficulty: MIT's current Kerberos version 5.5 β source-distribution [6] is not conveniently able to express the encryption-type of Alice's "hybrid" ticket. The MIT code currently assumes that a ticket has one encryption-type, which represents both the algorithm for decrypting the ticket, and the algorithm for which the ticket's session-key is intended. This is not a major problem, though, because this portion of the spec is too vague for other reasons, and needs to be reworked [14].

6 Conclusion

Our proposal judiciously uses the strengths of symmetric-key and public-key cryptography to solve the problems for which they are best suited. Most

[2] This hybrid use of Kerberos with RSA and a symmetric-key algorithm was proposed in [3].

Web users will get their Internet access from large corporate networks, so a natural corollary of their centralized account-admin is centralized key-distribution such as Kerberos provides. Web providers will be spread more thinly, so it makes sense that they should use public-key credentials. The proposal concentrates public-key encryptions at the servers, relieving clients of this substantial performance burden, and also relieving them of the responsibility for verifying certificates. Our proposal is compatible with the current specifications of the Kerberos system and the S-HTTP protocol.

7 Acknowledgements

Dan Geer and Jon Kamens at OpenVision, Win Treese at OpenMarket, Ted Ts'o at MIT, and the USENIX referees contributed helpful comments. Thank you all.

References

[1] D. Davis and D. Geer, "Kerberos Security With Clocks Adrift," *Proc. 5^{th} USENIX Security Symposium* (summer 1995).

[2] D. Davis and R. Swick, "Workstation Services and Kerberos Authentication at Project Athena," *LCS Technical Memorandum* **TM-424**, MIT Lab. for Comp. Sci. (February 1990).

[3] D. Davis and R. Swick, "Network Security via Private-Key Certificates," *USENIX 3^{rd} Security Symposium Proceedings*, (Baltimore; Sept. '92). Also in *ACM Operating Systems Review*, v.24, **4** (Oct. 1990).

[4] International Telegraph and Telephone Consultative Committee (CCITT). Recommendation X.509: The Directory - Authentication Framework. In *Data Communications Network Directory, Recommendations X.500-X.521*, pp. 48-81. Vol. 8, Fascicle 8.8 of *CCITT Blue Book*. Geneva: International Telecommunication Union, 1989.

[5] S.P. Miller, B.C. Neuman, J.I. Schiller, and J.H. Saltzer, *Project Athena Technical Plan*, Sec. E.2.1: "Kerberos Authentication and Authorization System," (Cambridge, Mass.) M.I.T. Project Athena internal document, Dec. 21, 1987.

[6] M.I.T. Information Systems, anonymous ftp distibution site for Kerberos software [athena-dist.mit.edu:pub/kerberos].

[7] Netscape Communications, "Secure Socket Layer Reference Document," Unofficial Internet Draft.

[8] C. Neuman and J. Kohl, *The Kerberos Network Authentication Service (V5)*, Internet RFC 1510, September 1993.

[9] C. Neuman, B. Tung, and J. Wray, "Public Key Cryptography for Initial Authentication in Kerberos," Internet Draft RFC, updates RFC 1510 (expires Sept. 1995).

[10] E. Rescorla and A. Schiffman, "Secure Hypertext Transfer Protocol," Internet Draft RFC (May '95).

[11] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, v. 21, **2**, Feb. '78, pp. 120-126.

[12] J.I. Schiller, D. Atkins, "Scaling the Web of Trust: Combining Kerberos and PGP to Provide Large Scale Authentication," *USENIX Winter Conference Proceedings*, January 1995.

[13] J.G. Steiner, C. Neuman, and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems", *USENIX Winter Conference Proceedings*, February 1988. [athena-dist.mit.edu:pub/kerberos/doc/usenix.PS]

[14] Ted Ts'o of MIT Information Systems, personal communication.

[15] P. Zimmermann, *Official PGP User's Guide*, Cambridge, Mass.:MIT Press, 1995.

Alliance Ecology: A Key to Multimedia Strategic Success

Bernard F. Mathaisel *Timothy S. Simcoe*

June 1995

Developments in multimedia are occurring faster than most companies can react to them. The industries that hope to create the content and control the infrastructure of the information superhighway are beginning to converge. Their convergence is driven by deregulation, global competition, the increasing complexity of products and services, and the accelerating pace of technological change. For many companies in these industries, strategic alliances have become a common way of acquiring new skills and accessing new markets. In a recent survey of more than one hundred executives from telephone, cable and other information carrying companies, seventy six percent of the respondents indicated that they believe "many or most" companies will need to enter a merger, alliance or partnership in order to survive. Trend watchers have begun to call the evolving networks of alliances "virtual corporations."

But there is strong evidence that alliances often fail to accomplish their intended goals. In another recent study of joint ventures done by a major consultancy, seventy percent either failed to meet their partners' expectations or were disbanded altogether. As alliances continue to gain importance, the ability to manage them successfully will be a strategic advantage. Companies with a set of alliance-related competencies will be able to quickly identify promising alliance opportunities and create maximum value in the relationships they enter. Alliance-savvy companies will also become attractive partners. As Rosabeth Kanter of the Harvard Business School has noted, "In the global economy today, companies are known by the company they keep."

As the number of alliances continues to grow, so will the number of ways which companies find to use them. But there is evidence that as the strategic ambition of an alliance grows, the art of managing it for maximum value becomes more and more complex. *In this paper, we draw on the principles of ecology and an examination of complex natural systems to identify lessons which can be applied to managing complex and strategically ambitious alliances.*

Types of Alliances

Alliances are not a new idea. Cartels, which attempt to suppress open competition, have existed for hundreds of years. In recent years much attention has also been paid to Japanese *keiretsu*, groups of large enterprises which come from diverse industries and have long standing and broad based cooperative arrangements. The alliances we examine are different from both cartels and *keiretsu*. Cartels are inter-industry organizations while strategic alliances are built between a number of related industries. As a result, alliances contain more inter- and intra- industry competition than cartels. And compared to *keiretsu,* strategic alliances' company roles are more narrow and more strategically focused.

Characterizing alliances can be difficult because of the number of forms they take, ranging from loose cooperative agreements or consortium to jointly owned ventures under strict control. There are also several ways of classifying alliances - whether along the established axes of vertical and horizontal integration, according to the participants' degree of similarity, or using the structural and legal attributes of the deal. The most common classification is a structural one which differentiates among three types of alliance by focusing on rights, ownership, and organization. The three types are:

- *Joint Ventures*: Joint ventures are formed by two separate entities under a new and separate management umbrella. They may be created by combining existing assets or jointly investing in new ones. They may operate independently of the parents or as a subsidiary. Joint ventures allow parent firms to maintain their identity while sharing risks and rewards with their subsidiary.

- *Equity Partnerships:* Equity partnerships involve the purchase of a minority investment in one firm or an agreement in which two firms purchase shares in each other. A new legal entity is formed, but it is not under separate management and it is governed according to the strategic understanding developed between the two parties.

- *Franchises/Licenses:* Franchises or licenses may qualify as alliances if their intent is strategic. These agreements grant the license holder specific rights to products, technology, business assistance, distribution or branding. For these agreements to constitute an alliance there must be a degree of joint commitment and shared risk.

Alliance Necessity

There are a number of reasons why companies enter into alliances. The principle drivers of alliance formation are:

- *Combining Knowledge - Alliances can assemble teams that have a combined understanding of technology, industry, processes and management.* Example: Bell Atlantic has allied with Intel, Gandalf, J&L, 3COM and Ernst & Young to provide solutions which integrate the hardware, software and work process redesign elements of the problem of transitioning to a tele-commuting workplace.

- *New Competencies - Alliances can provide access to new and unfamiliar skills in the context of a relationship which is less "arms-length" than contracting or outsourcing.* Example: Time Warner allied with AT&T and Silicon Graphics, among others, to acquire the technical skills necessary for their Full Service Network trial in Orlando.

- *Market Identity and Relationships - Entering an alliance is a quick way to establish an identity and build relationships in a new market.* Example: Many retail firms are trying to establish a presence in the emerging market for electronic commerce through alliances with America Online and other service providers.

- *Barriers to Entry - Alliances are used to "jump" barriers to entry (e.g. regulatory) in new geographic or product markets.* Example: AT&T-McCaw Cellular and MCI-Nextel are deals which position long distance carriers to hop the regulatory barrier and provide local telephone service.

- *Standards Agreements - Alliances pool the strength of companies which have a common interest in the outcome of industry standards disputes.* Example: Compaq and Microsoft have an alliance in which they are trying to develop video server standards around PC based technology.

- *R&D Scale Economies - By sharing the costs of developing a new technology, alliances accelerate return on investment.* Example: Apple, IBM and Motorola formed a consortium to develop the PowerPC chip.

- *Culture Incubation - Alliances provide a vessel in which a culture separate from that of the parent companies can develop.* Example: In a joint venture to develop automation software with Rensselaer Polytechnic Institute, GE sent its managers to a "skunk works" at the college campus to foster an atmosphere of collegiate informality.

- *Risk Management - Litigation risk is reduced as companies distinguish alliance activities from the parent.* Example: The creation of Dow-Corning, a subsidiary joint venture, helped shelter both Dow Chemical and Corning from costly litigation over silicone breast implants.

Strategic Ambition

Strategic alignment and shifts in strategy are an important factor to consider when forming an alliance. Partners move in different directions, priorities change, and therefore partner terms and arrangements that worked at one time can become less satisfactory. Companies need to define individual and collective objectives clearly before trying to create a strategy for their alliance.

In Partnerships for Profit Jordan Lewis gives an example of a good alliance with limited strategic objectives. Two fierce competitors in the hardware market, Apple and Digital, formed an alliance in order to compete against IBM. Both companies, however, agreed *only* to develop and adhere to common interface standards. Lewis points out that the alliance's success was linked to the limited level of cooperation. As multimedia alliances proliferate, being able to assess the appropriate level of cooperation and tailor the strategic aims of the alliance to that level will be important.

Ernst and Young has developed a framework for examining the kinds of strategy that are emerging in the field of multimedia. The framework measures the impact of technological change on the supply and the demand side of a business, and maps the impact level to one of three kinds of strategy:

- *Improve Existing Processes*
- *Restructure Industry*
- *Develop New Industry*

At each new level the strategy is more ambitious, because new technology impacts the industry more dramatically. As firms move towards developing new industries the payoffs increase, the challenges become larger and the risks grow. For many of the reasons discussed earlier, companies have found that entering strategic alliances can help them pursue a more ambitious strategy.

But greater strategic ambition leads to greater complexity within the alliance relationship. In Winning Combinations, James Botkin and Jana Matthews quote a Kodak manager on trying to create "entrepreneurial alliances" within a large corporation, "... it was a little bit like trying to create a free enterprise system within a planned economy." More ambition leads to more cooperation, which leads to the involvement of more people, organizations and markets, and above all, the creation of more connections between them. Establishing and maintaining control can rapidly become an extremely difficult problem. But instituting a "planned economy" can stifle the overriding objective of creating a new industry. Ecology, the study of complex natural systems, offers some answers to this management dilemma. As the strategic ambition of an alliance increases, so must the level of alliance ecology.

Alliance Ecology

Much of the existing work on alliance formation and management already hints at the complexity of alliance relationships and their parallel to natural systems. For example, Rosabeth Kanter writes, "[Alliances] are living systems that evolve progressively in their capabilities... they cannot be controlled by formal systems but require a dense web of interpersonal connections." But, to our knowledge, no one has explicitly extended this metaphor by jointly examining alliances and ecology.

In the book Filters Against Folly, Garrett Hardin applied the constructs of ecological thinking to social and other non-biological problems. He emphasized that the "ecolate filter" offers us a new level of understanding. Applying Hardin's three filter framework to alliances will help us begin to define the concept of "ecological thinking."

Hardin's Three Filter Framework

Filter	Expressed	Form	Characteristics
Literate	Written Word	Contracts	Rigid
Numerate	Numbers	Gain sharing	Subject to Gaming
Ecolate	Relationships	Systemic View	Organic

A "literate filter" can be likened to understanding an alliance's potential synergies. The "literate" expression of two firms' mutual understanding is a contract. While contractual understanding is necessary, useful and legally recognized, it can be rigid. A "numerate filter" is the result of using analytical tools to examine the state of the alliance. A numerate understanding is expressed in the form of the parents' gains sharing agreement. Analytical tools are extremely valuable filters of information and can be applied throughout an enterprise, from balance sheets to performance measurements. But a numerate filter fails to capture anything that is difficult to quantify and can be extremely sensitive to numbers which are subject to gaming.

An "ecolate filter" looks beyond the rationale for the deal or its legal structure. The ecolate filter, like the literate or numerate filter, recognizes the importance of the relationship between the partners in an alliance. But in keeping with the holistic view of ecology, the ecolate filter also emphasizes the relationship *between the alliance and its environment.* It forces managers to ask questions such as: How will the fruits of the relationship change its initial mission? Will the parents' distinctive competencies remain equal in strength yet complementary as the alliance matures? and How can the venture expand into a new industry while maintaining flexibility and accountability to the partners? The expression of an "ecolate" perspective is a solid, yet organic, relationship which extends "beyond the deal."

Much evidence already exists to support the view that it is important to look "beyond the deal" and place more emphasis on the "whole system" in which an alliance operates.

- Katherine Harrigan, Strategies for Joint Venture Success: "Managers sometimes believed erroneously that they could set up joint ventures and let them run themselves. Most joint ventures required much more management time than many parents had expected."

- Rosabeth Kanter, "Collaborative Advantage" in Harvard Business Review August 1995: "The romance of courtship quickly gives way to day-to-day reality as partners begin to live together." (She goes on to detail many of the problems of broader involvement as well as ways for dealing with the inevitable discovery of differences in an alliance.)

- Jordan Lewis, Partnerships for Profit: "Executives with successful [alliance] experiences kept redirecting our interviews to how people on both sides of their alliances had built strong relationships. For them, strategic gain was the motivation for alliances, yet strategy was an intellectual construct that worked or didn't depending on relationships."

Managing Ecologically

Hardin's three filter framework helps illustrate the ecological perspective, and it points out some of the dangers of *not* thinking ecologically. However, it does not offer any specific advice on managing or controlling a complex alliance. Developing a set of guidelines for ecological management is the next step. In order to sharpen our focus and arrive at some managerial guidelines, we will use a small but broad set of "ecological principles" to simplify our thinking. The principles were first developed to study biological systems, but they are useful in analyzing any complex system operating in a rapidly changing environment.

1) *Emphasize the relationship between organism and environment*
   - *Difficulty of drawing hard boundaries within the system*
   - *"Whole" or "soft" systems perspective*
2) *Adapt to a state of "dynamic equilibrium"*
   - *Effect of "feedback loops"*
   - *New strategies for intervention and change management*

The key to the first principle is a good understanding of the word ***environment***. Seeing the environment as a "whole" system without hard boundaries does not come naturally to most people. We have been taught to reduce problems to a manageable size by making simplifying assumptions. The assumptions can make problems tractable, but as systems become more complex the same assumptions often miss subtle yet important influences.

Aldo Leopold, one of the fathers of ecology, attempted in 1934 to recreate a natural prairie on a piece of purchased farmland. For a number of years the project met with only limited success under his stewardship. The experiment only began to flourish after an unexpected ingredient, one which the ecologist had been guarding against, was introduced. The ingredient was fire, and it caused the seeds of many prairie plants to be released and spread while it wiped out many urban predators. It is a good example of how the simplifying assumptions we make, while seemingly perfectly reasonable, may prevent us from seeing important connections within a complex system. Here are some ecological rules of thumb that can help an alliance develop and maintain a strategy which does not lose sight of the whole environment in which it operates.

- *Establish a general vision*
- *Set goals at the "macro" level*
- *Constantly reevaluate your mission*
- *Avoid short-term highly specific numeric targets*
- *Build relationships at multiple levels*
- *Pay attention to stimuli*

The second ecological principle we focus on involves "***dynamic*** equilibrium." The competitive environment, like the natural environment, is in constant flux. Adaptability is important, but gaining it requires giving up a degree of control. The ability to manage change is a key asset for any alliance which hopes to evolve, but managing change in even a simple natural system has proven extremely difficult. The solution involves letting change begin at the bottom rather than the top and focusing energy on assuring that low level processes are working smoothly.

Rodney Brooks, an engineering professor at MIT, tries to make "unintelligent" robots perform complex tasks. He has found that the way to do this is to distribute the control of higher level functions among subsystems which handle separate jobs. "Brooks' law" can be stated "Complexity must be grown from simple systems that already work." His lab has developed a generic recipe for distributed control which generalizes to the task of managing and evolving any complex system such as an ambitious strategic alliance.

- *Do simple things first*
- *Learn to do them flawlessly*
- *Add new activities over the results of the simple*
- *Don't change the simple things*
- *Make new things work as flawlessly as the simple*
- *Repeat*

It is difficult to measure or even provide examples of alliance success which is directly attributable to our ecological principles. The task is made difficult by the problem of defining "alliance success" , the lack of "inside information" on alliance management and the difficulty of quantifying our ecological insights. There is, however, a great deal of circumstantial evidence supporting our principles of ecological management. Much of the literature on alliances, mergers and joint ventures focuses on relationships, whether firm-to-firm or alliance-to-environment. Like our first principle, these papers emphasize cooperation on multiple levels, setting very general high level goals and a willingness to nurture the alliance in the hope of unexpected benefits. There are also a variety of case studies on customer-supplier relationships, outsourcing, and new "Japanese" manufacturing processes that detail the benefits of distributing responsibility and control, layering complex processes and focusing on making low-level processes run smoothly. These are essentially the components of the second principle. Part of the conclusion of this research project will be an attempt to collect stories of successful applications of ecological management in multimedia alliances (perhaps through an appeal to several usenet newsgroups).

Ecological Lessons

Many alliances fail because companies try to architect them, instituting formal rather than distributed control, insisting on hard boundaries between the parents and the alliance, and focusing on the structure of the deal rather than the relationships involved. The three filters of Garrett Hardin, our short list of ecological principles and the strategic and control rules for complex systems are the building blocks of a different way of thinking about managing alliances. We can use them to begin to develop a list of best practices for alliance management.

- *Emphasize the **relationship** between companies.* A deal may look great on paper or when high level executives meet, but if the *relationship* is not good at a political, cultural, organizational or human level, the alliance is in trouble. Devote more, rather than fewer, people to managing relationships, and empower relationship managers to vary their own company procedures. Work to build trust by anticipating cultural differences and conflicts of loyalty.

- *Don't draw hard boundaries within the alliance.* Successful alliances require collaboration and not merely exchange. Strategic, tactical, operational, cultural, and inter-personal integration all must take place. Create a good infrastructure for learning which includes cross-functional teams and widespread information sharing. This will help to prevent a dangerous imbalance of information resources between firms.

- *Establish a general vision and maintain broad goals* - Alliances need room to evolve. Mission statements and well defined goals promote direction, but it is important to avoid measuring an alliances progress using inflexible numeric targets as goals.

- *Develop new strategies for intervention and change management.* The ability to adapt to changes in the competitive environment is crucial to the long term success of an alliance. Develop processes which promote continuous change and which keep the alliance "close" to

its markets by distributing responsibility and control while assuring that low-level processes are working.

- *Don't think of anything as a side effect* - Many of the most valuable new markets, processes, technologies and cultural changes to come out of alliances are unexpected. Encourage experimentation, and cross-fertilization in a wide-open R&D environment.

- *Manage all your alliances* - The value of a holistic approach applies to alliance networks as well as alliances. A paper by Benjamin Gomes-Casseres "Group versus Group: How alliance Networks Compete" and a report by the Conference Board, "Strategic Alliances: Guidelines for Successful Management" both emphasize the importance of actively managing a firm's entire portfolio of alliances as well as tending to them on an individual basis.

Conclusion

The strategic ambitions of an alliance will determine the difficulty of managing it. Many firms, in keeping with their limited ambitions, will choose to limit the scope of their alliances. But alliances born with the ambitious goal of creating dramatic changes in multimedia industries will need to be managed ecologically.

Ecologists have observed that rapid change, flexibility and adaptability are common to large and complex natural systems. Yet it has proven difficult for large complex systems such as hierarchical companies which are organized around huge economies of scale to achieve those qualities. The creation of new markets and industries is often accomplished by small start-ups: young, high-growth companies which are close to their markets and are not only flexible, but are committed to actually creating change. These companies focus on their relationship to the changing market environment, refuse to draw hard boundaries, develop new ways of dealing with change, and capitalize on unexpected opportunities. As multimedia evolves, ambitious strategic alliances and alliance networks will try to achieve the small business mentality. Ecological thinking will be the key.

A recent survey of communications executives concluded that, "The keys to success... are the speed, flexibility, and focus of a small business made large by a network of knowledge. No matter how large the [successful] enterprise... it will resemble a collection of small businesses." Ecological thinking can help create integrated alliances which think like small companies but have the resources of large ones. Those alliances are the ones which promise not only to survive, but to thrive, in the interactive future.

Bibliography

Botkin, James and Lana Matthews: Winning Combinations, John Wiley & Sons Inc., 1992.

Gates, Stephen: Strategic Alliances: Guidelines for Successful Management, The Conference Board Inc., 1993.

Gomes-Casseres, Benjamin: "Group Versus Group: How Alliance Networks Compete", Harvard Business Review, July-August 1994.

Hardin, Garrett: Filters Against Folly, Penguin Books, 1985.

Harrigan, Katherine: Strategies for Joint Ventures, Lexington Books, 1985.

Kanter, Elizabeth: "Collaborative Advantage", Harvard Business Review, July-August 1994.

Kelly, Kevin: Out of Control, Addison Wesley, 1994.

Lewis, Jordan: Partnerships for Profit, Macmillan, 1990.

Maney, Kevin: Megamedia Shakeout, John Wiley & Sons Inc., 1995.

Myers, Paul: Managing For Joint Venture Success, forthcoming, Ernst & Young.

Troy, Katheryn: Change Management: Strategic Alliances, The Conference Board Inc., 1994.

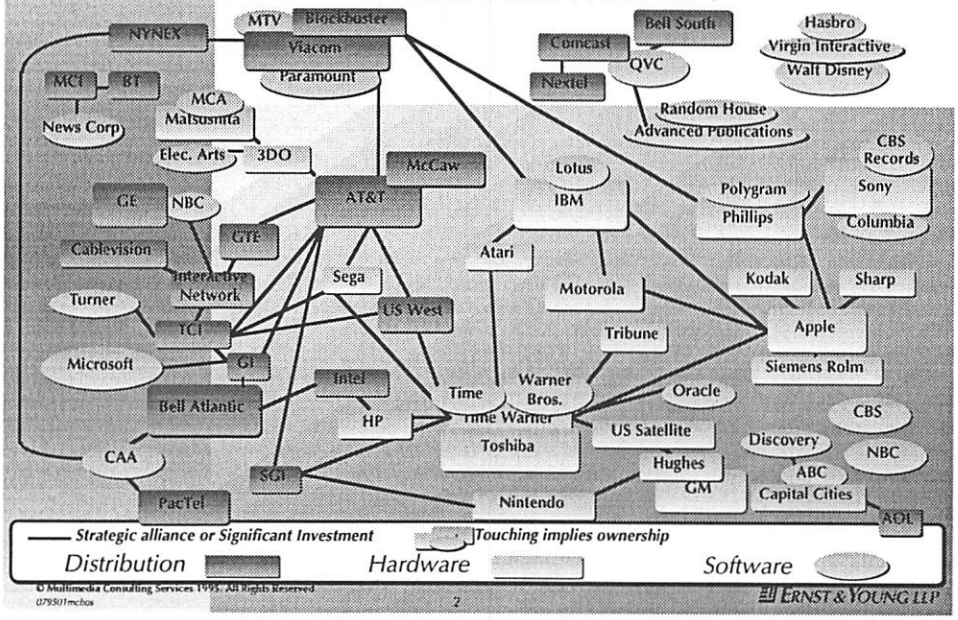
Alliance Ecology

A Key to Strategic Success in Electronic Commerce

Bernard F. Mathaisel
Director, Multimedia Consulting Services
Ernst & Young LLP

July 12, 1995

An Overview of the Convergence of the Global Multimedia Market



Electronic Commerce Demand and Supply Impact

Emergence of Electronic Commerce

Potential Impact on *Demand* Drivers

How could these technologies change . . .

1. the basic needs of that decisionmaker?
2. who makes/influences the decision?
3. the decision making criteria used?
4. how information about these criteria is obtained?

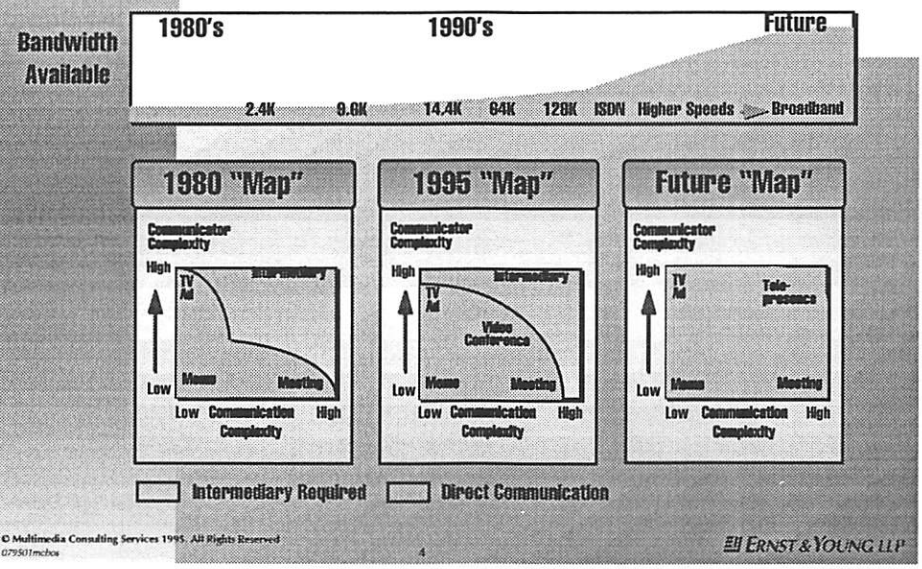
Potential Impact on *Supply* Drivers

How could these technologies change . . .

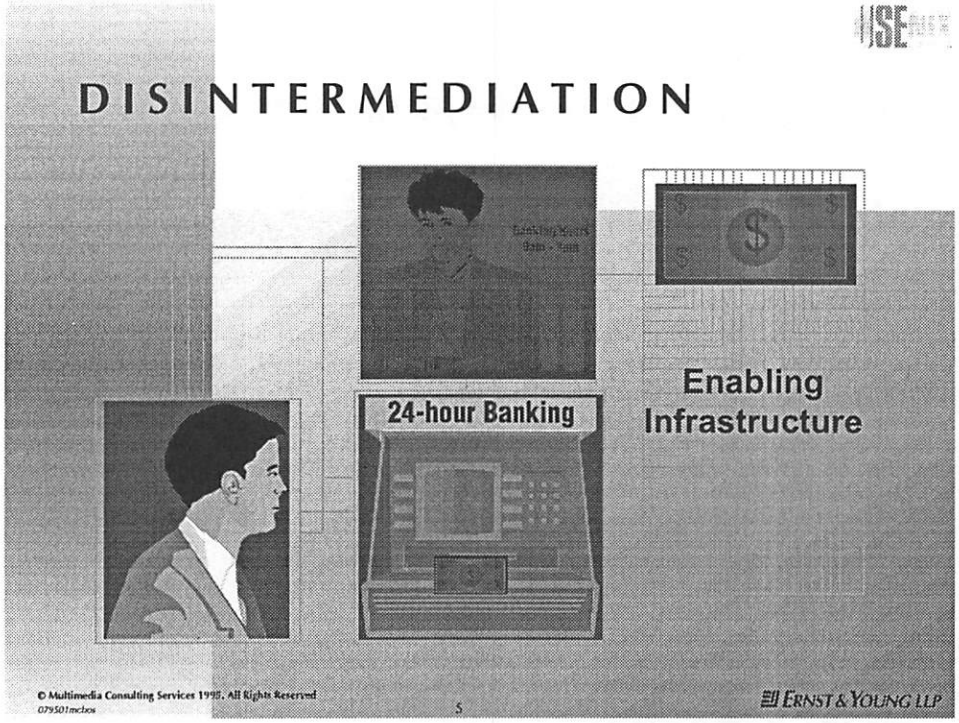
1. the supply chain?
2. internal effectiveness?
3. production economics?
4. industry "power" relationships?

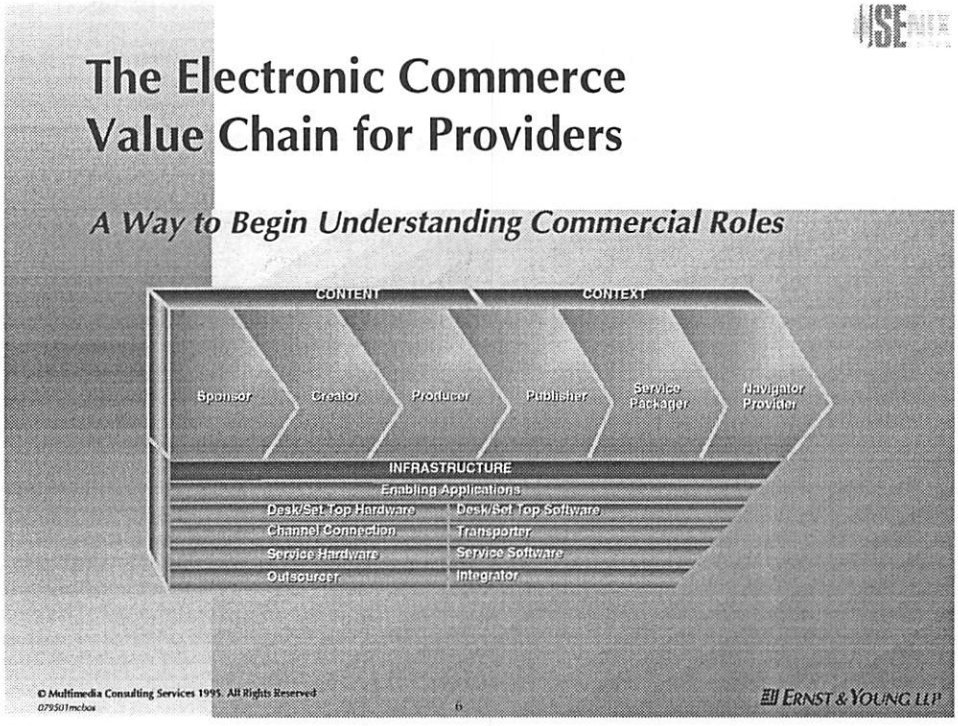
ERNST & YOUNG LLP

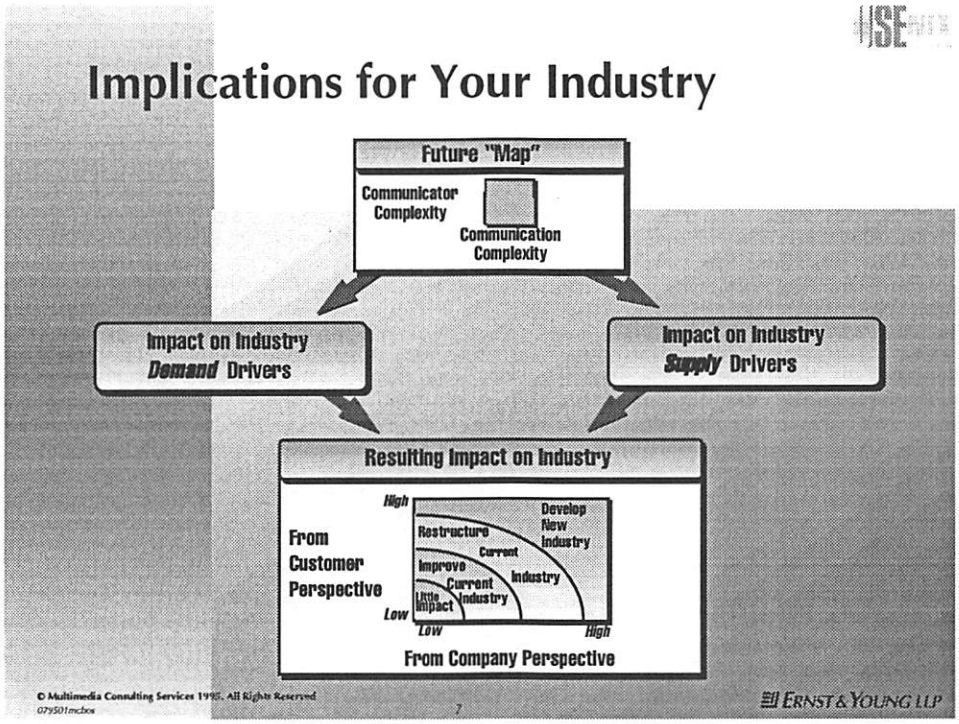
Changing Maps

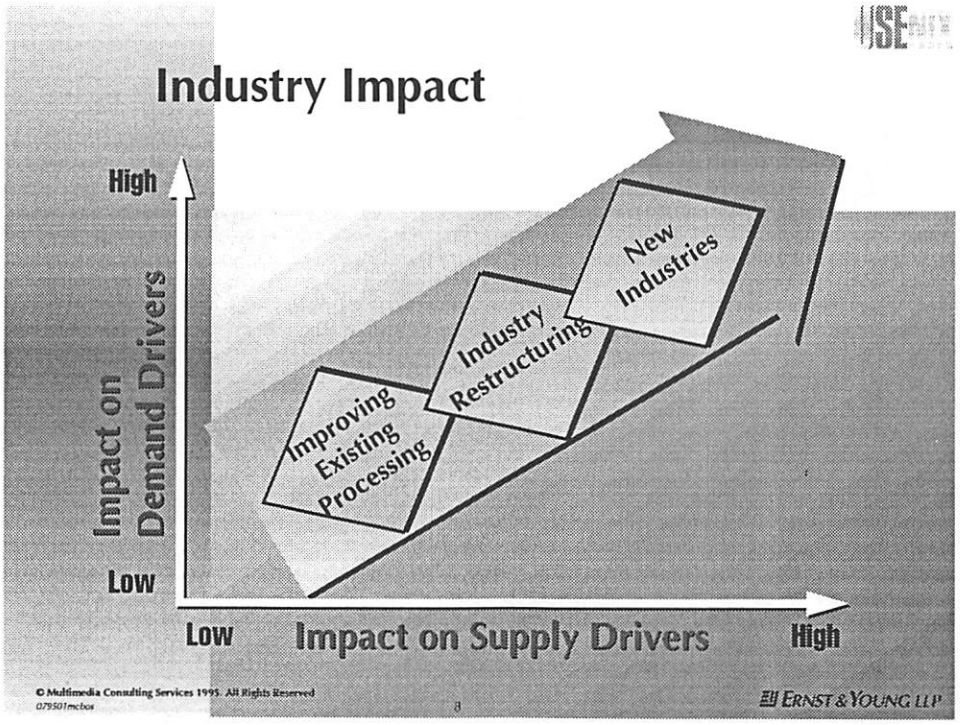


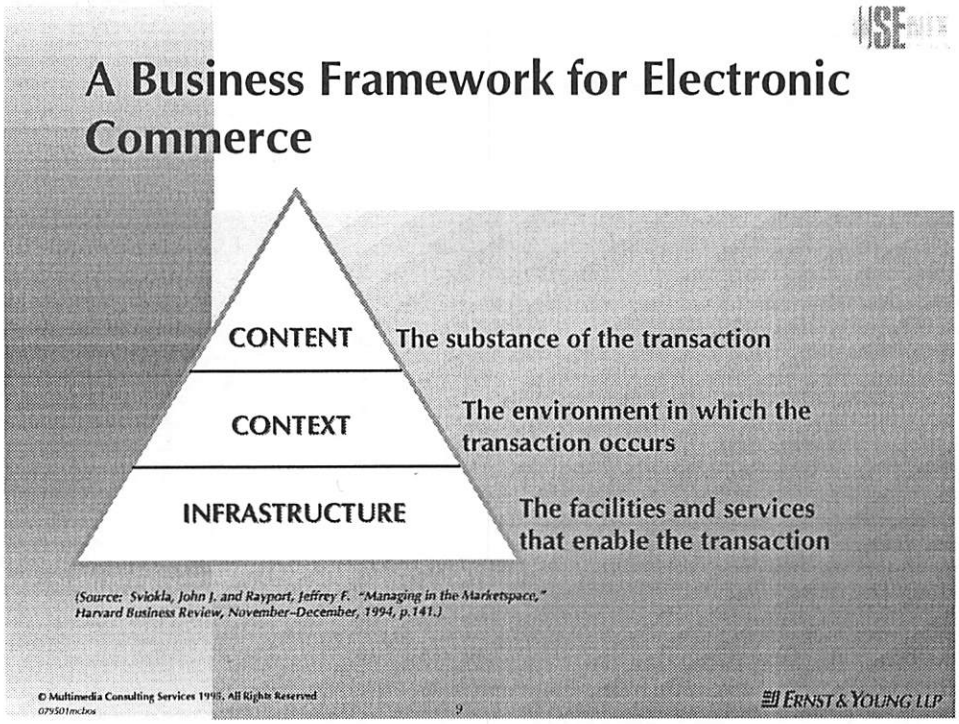
ERNST & YOUNG LLP

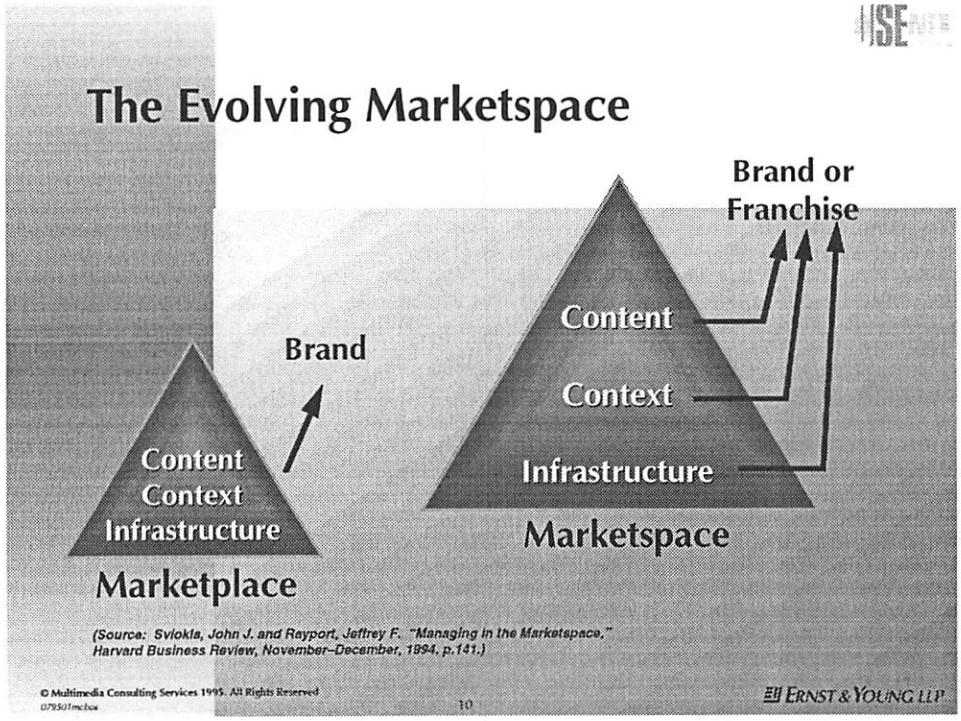

DISINTERMEDIATION
24-hour Banking
Enabling
Infrastructure
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
5
ERNST & YOUNG LLP

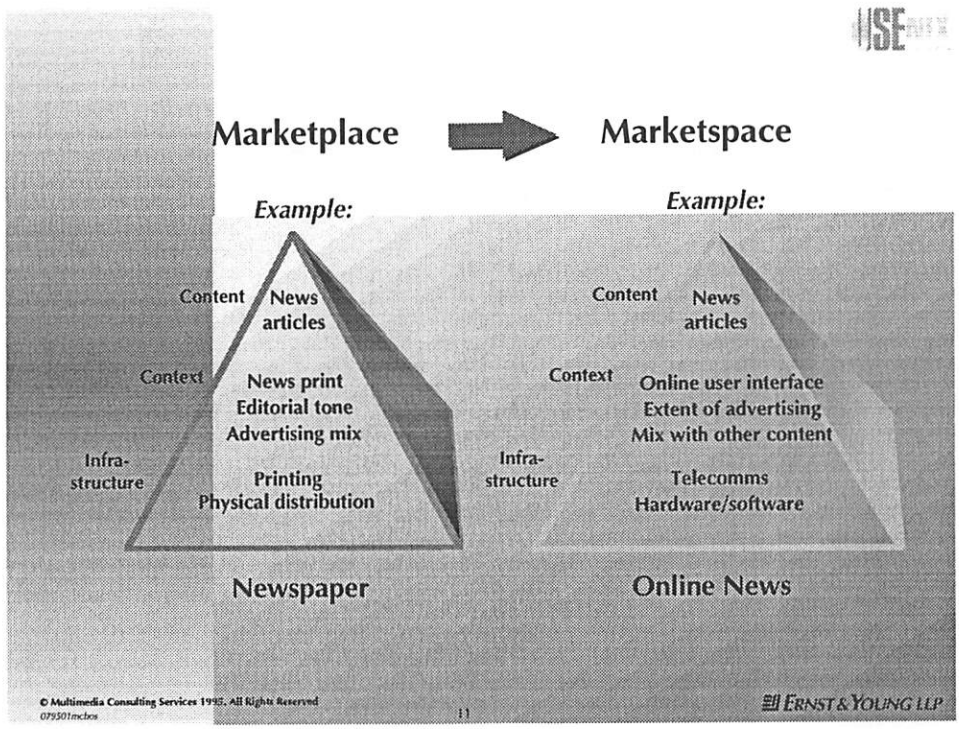

The Electronic Commerce
Value Chain for Providers
A Way to Begin Understanding Commercial Roles
CONTENT
CONTEXT
Sponsor
Creator
Producer
Publisher
Service Packager
Navigator Provider
INFRASTRUCTURE
Enabling Applications
Desk/Set Top Hardware
Desk/Set Top Software
Channel Connection
Transporter
Service Hardware
Service Software
Outsourcer
Integrator
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
6
ERNST & YOUNG LLP

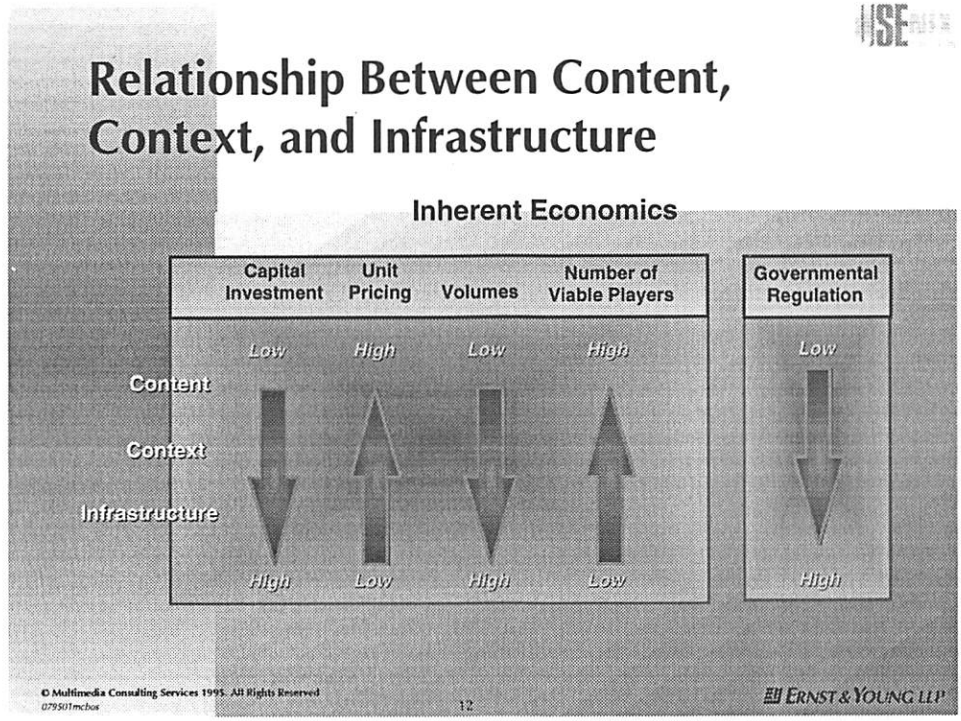

Implications for Your Industry
Future "Map"
Communicator Complexity
Communication Complexity
Impact on Industry Demand Drivers
Impact on Industry Supply Drivers
Resulting Impact on Industry
From Customer Perspective
High
Low
Develop New Industry
Restructure Current Industry
Improve Current Industry
Little Impact
Low
High
From Company Perspective
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
7
ERNST & YOUNG LLP

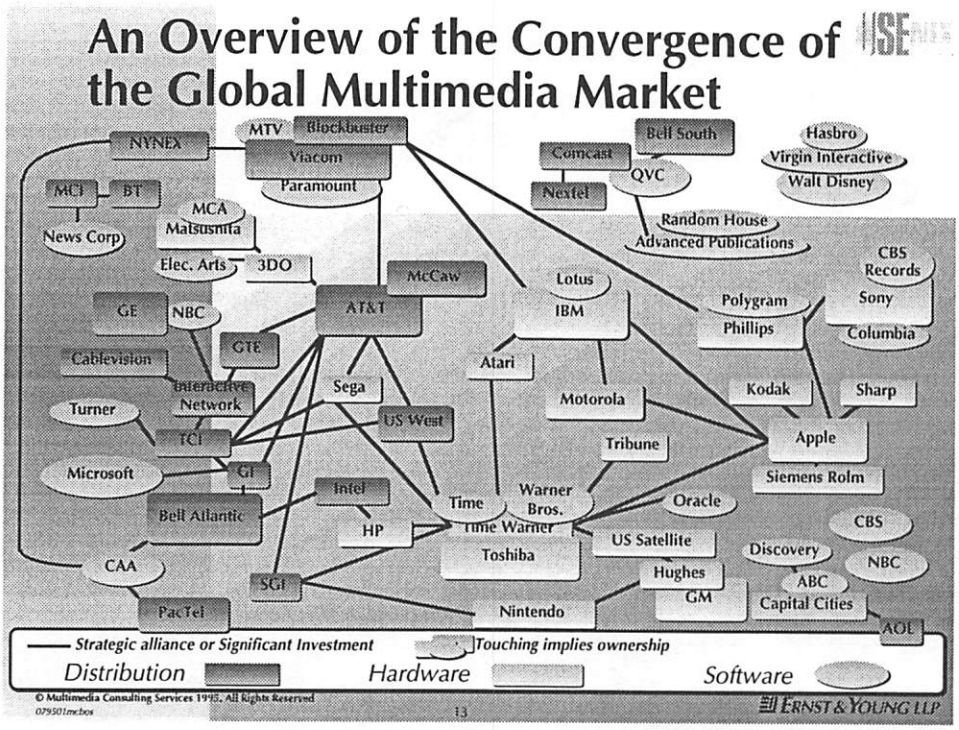

Industry Impact
High
Impact on Demand Drivers
Low
New Industries
Industry Restructuring
Improving Existing Processing
Low
Impact on Supply Drivers
High
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
8
ERNST & YOUNG LLP


A Business Framework for Electronic Commerce
CONTENT
The substance of the transaction
CONTEXT
The environment in which the transaction occurs
INFRASTRUCTURE
The facilities and services that enable the transaction
(Source: Sviokla, John J. and Rayport, Jeffrey F. "Managing in the Marketspace," Harvard Business Review, November–December, 1994, p.141.)
© Multimedia Consulting Services 1995. All Rights Reserved
9
ERNST & YOUNG LLP


The Evolving Marketspace
Brand or Franchise
Content
Context
Infrastructure
Marketspace
Brand
Content
Context
Infrastructure
Marketplace
(Source: Sviokla, John J. and Rayport, Jeffrey F. "Managing in the Marketspace," Harvard Business Review, November–December, 1994, p.141.)
© Multimedia Consulting Services 1995. All Rights Reserved
10
ERNST & YOUNG LLP

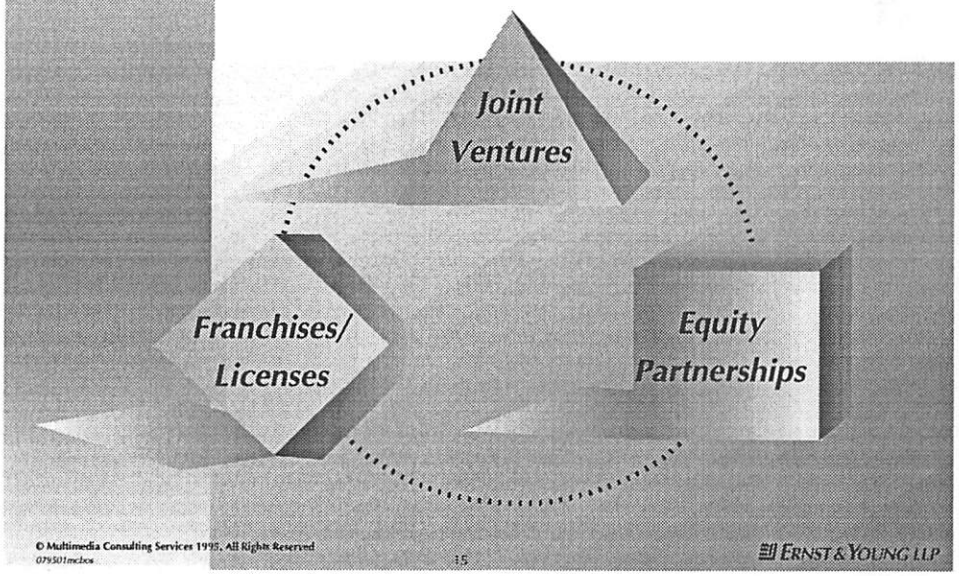

Marketplace
Marketspace
Example:
Example:
Content
News articles
Context
News print
Editorial tone
Advertising mix
Infra-structure
Printing
Physical distribution
Content
News articles
Context
Online user interface
Extent of advertising
Mix with other content
Infra-structure
Telecomms
Hardware/software
Newspaper
Online News
© Multimedia Consulting Services 1995. All Rights Reserved
079501mchos
11
ERNST & YOUNG LLP


Relationship Between Content, Context, and Infrastructure
Inherent Economics
Capital Investment
Unit Pricing
Volumes
Number of Viable Players
Governmental Regulation
Low
High
Low
High
Low
Content
Context
Infrastructure
High
Low
High
Low
High
© Multimedia Consulting Services 1995. All Rights Reserved
079501mchos
12
ERNST & YOUNG LLP


An Overview of the Convergence of the Global Multimedia Market
NYNEX
MTV
Blockbuster
Viacom
Paramount
Bell South
Comcast
QVC
Nextel
Hasbro
Virgin Interactive
Walt Disney
MCI
BT
MCA
Matsusmita
News Corp
Random House
Advanced Publications
Elec. Arts
3DO
McCaw
Lotus
CBS Records
GE
NBC
AT&T
IBM
Polygram
Phillips
Sony
Columbia
GTE
Cablevision
Atari
Interactive Network
Sega
Motorola
Kodak
Sharp
Turner
US West
TCI
Tribune
Apple
Microsoft
GI
Siemens Rolm
Intel
Time Warner
Bros.
Oracle
Bell Atlantic
HP
CBS
US Satellite
Discovery
Toshiba
NBC
CAA
Hughes
ABC
SGI
GM
Capital Cities
PacTel
Nintendo
AOL
Strategic alliance or Significant Investment
Touching implies ownership
Distribution
Hardware
Software
© Multimedia Consulting Services 1995. All Rights Reserved
079501mchos
13
ERNST & YOUNG LLP


Reasons for Alliances
Combining Knowledge
New Competencies
Market Identity and Relationships
Barriers to Entry
Standards Concurrence
R&D Scale Economies
Cultural Incubation
Risk Management
© Multimedia Consulting Services 1995. All Rights Reserved
079501mchos
14
ERNST & YOUNG LLP

Types of Alliances



15

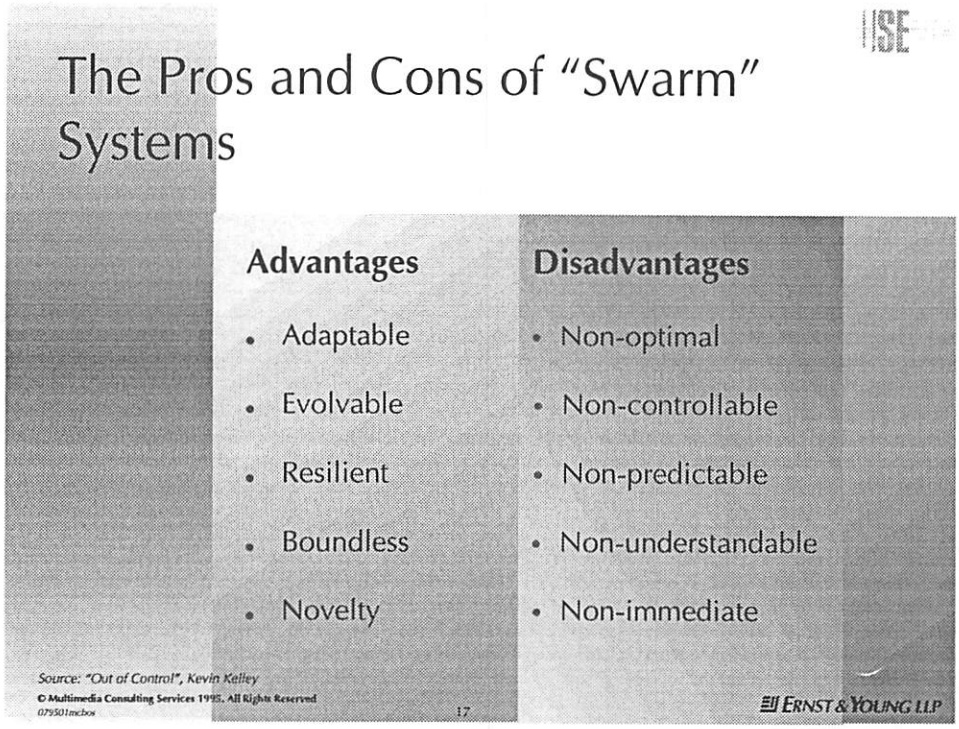
ERNST & YOUNG LLP

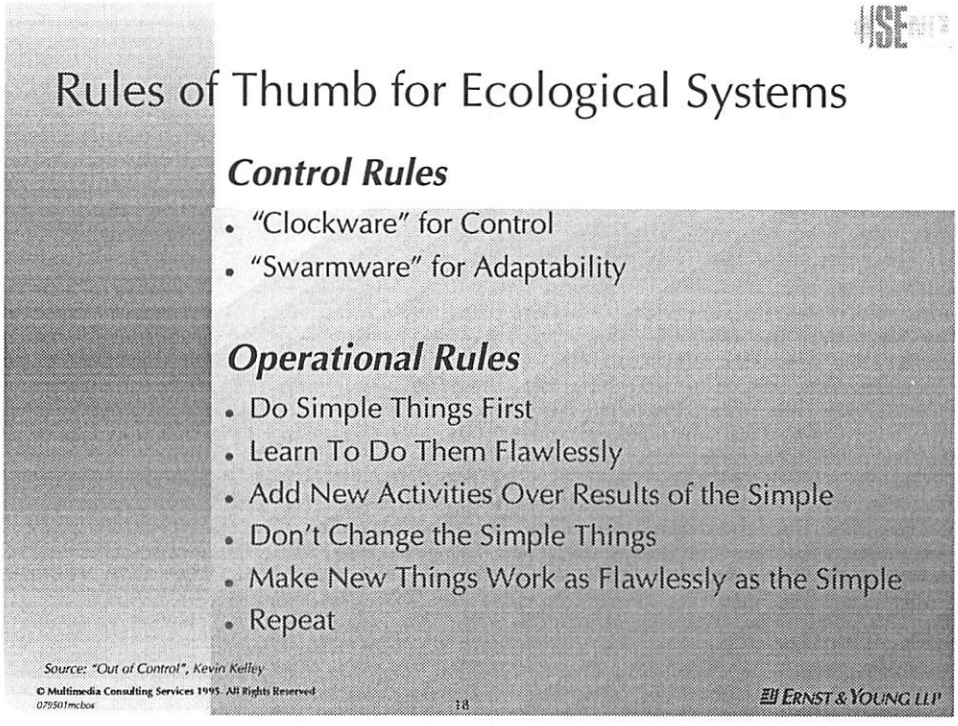
"Filters Against Folly"

Level	Expressed	Form	Characteristics
Literacy	Written Word	Contracts	Rigid
Numeracy	Numbers	Gain Sharing	Subject to Gaming
Ecolacy	Systemic View	Relationships	Organic

Source: "Filters Against Folly", Garrett Hardin, Penguin Books, © 1985

ERNST & YOUNG LLP

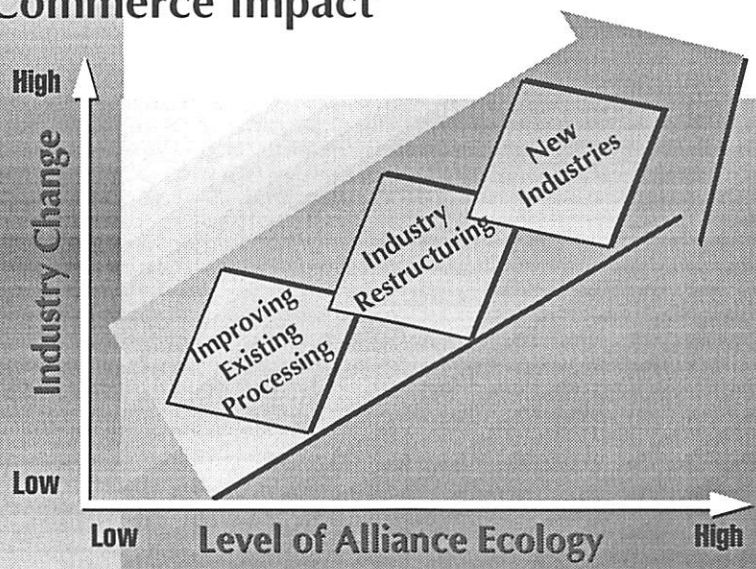

The Pros and Cons of "Swarm" Systems
Advantages
• Adaptable
• Evolvable
• Resilient
• Boundless
• Novelty
Disadvantages
• Non-optimal
• Non-controllable
• Non-predictable
• Non-understandable
• Non-immediate
Source: "Out of Control", Kevin Kelley
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
17
ERNST & YOUNG LLP

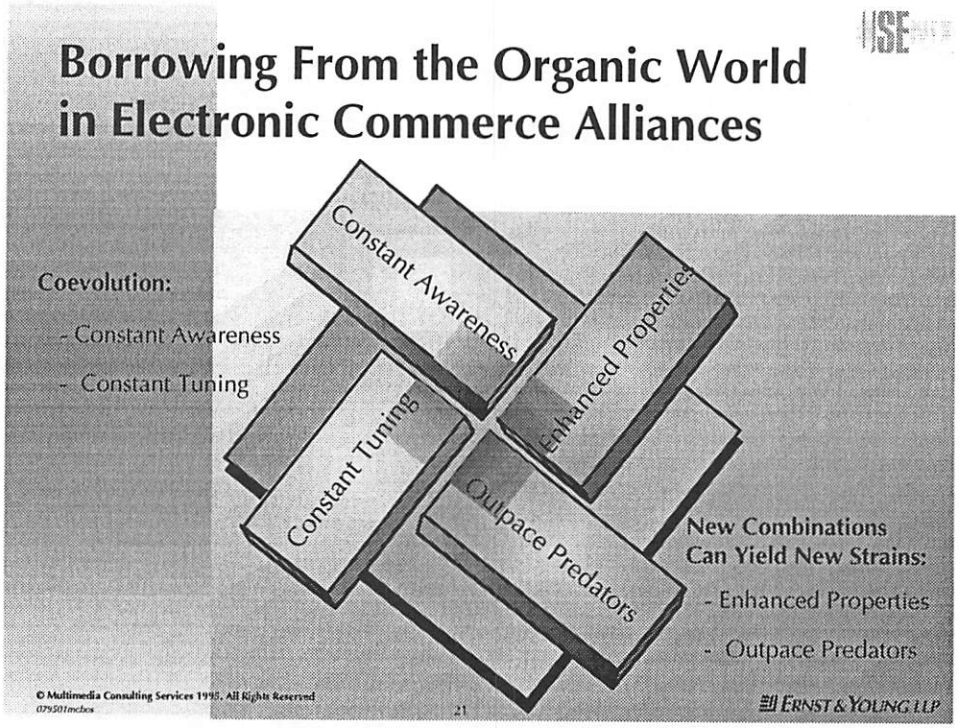

Rules of Thumb for Ecological Systems
Control Rules
• "Clockware" for Control
• "Swarmware" for Adaptability
Operational Rules
• Do Simple Things First
• Learn To Do Them Flawlessly
• Add New Activities Over Results of the Simple
• Don't Change the Simple Things
• Make New Things Work as Flawlessly as the Simple
• Repeat
Source: "Out of Control", Kevin Kelley
© Multimedia Consulting Services 1995. All Rights Reserved
079501mcbos
18
ERNST & YOUNG LLP

The Value of an Ecological Perspective

- **Emphasize the relationship between organisms and their environment**
  - *Difficult to draw hard boundaries within the system*
  - *Need to adopt a "whole" or "soft" systems perspective*
- **"Coevolution"**
  - The effects of co-dependence and feedback loops
  - New approaches for intervention and change management

Alliance Ecology and Electronic Commerce Impact




Borrowing From the Organic World in Electronic Commerce Alliances
Coevolution:
- Constant Awareness
- Constant Tuning
Constant Awareness
Enhanced Properties
Constant Tuning
Outpace Predators
New Combinations Can Yield New Strains:
- Enhanced Properties
- Outpace Predators
© Multimedia Consulting Services 1995. All Rights Reserved
21
ERNST & YOUNG LLP

Digest
USENIX Workshop on Electronic Commerce
July 11–12, 1995

P. Honeyman
honey@citi.umich.edu

ABSTRACT

This two-day meeting was organized to foster interaction between the commercial and technical communities engaged in the design and deployment of systems of electronic commerce. Attendance was by invitation from among those who responded to the call for participation. The workshop featured lightly refereed formal presentations, panel sessions, invited talks, and broad, open-ended discussions on technological and social topics.

1. Tuesday, July 11, 1995

Ninety-four attendees were invited to the USENIX Workshop on Electronic Commerce. In his opening remarks, Program Chair Dan Geer set the tone for the conference by requesting that participation be spirited, stressing the urgency of solving electronic commerce problems before the "cumulative investment" of the Internet, more commercial than ever, pins down policies that might wreak havoc with privacy, security, and needed functionality.

1.1. Keynote address

Electronic Banking and Electronic Commerce on the SuperInformation Highway
Daniel Schutzer
Citibank

Dan Schutzer is Vice President and Director of Advanced Technology at Citibank, and President of the Financial Services Technology Consortium. Dan opened his keynote address with an overview of the present information infrastructure for electronic commerce, consisting of the combination of a number of components (*e.g.*, the Internet, telephones, EDI, *etc.*). Continuing to develop at a rapid pace, the resultant information infrastructure may revolutionize the way commerce is conducted. Components are provided by several different industries that need to integrate their products and divide the costs and profits from the resultant infrastructure.

Dan sees both excitement and fear about achieving wide-scale electronic commerce. Electronic commerce has the potential to break monolithic corporations into smaller, more efficient organizations that buy and sell from each other via the net. Because so much of the net is free, a shift will result in the value-chain between producer and consumer. For example, the net blurs distinction of place, making it no more or less efficient to go inside or outside the company, so in-house design or production departments might bid on contracts just like outsiders.

Electronic commerce involves everything one can do in the physical world: advertising, shopping, bartering, negotiating contracts and prices, bidding for contracts, ordering, billing, payment, settlement, accounting, loans, bonding, escrow, *etc.* Electronic commerce is both a threat and a promise — it holds the promise of new markets, channels, types of products, and employment, but threatens present jobs and personal security. To reduce these fears, the information infrastructure should provide privacy, protection of intellectual property rights, authentication, authorization, non-refutability, and reliable payment structures.

At the heart of electronic commerce is an act of payment; the information infrastructure implements electronic commerce primarily by implementing that act. Payment can take place in several ways, and many payment methods must be supported. At present, there are lots of players and protocols. Interoperability among different payment systems is critical. Without it, fears about the value of different electronic currencies will scare customers away. With it, the payment process

can be automated to reduce costs, which, along with ubiquity of service, can improve customer satisfaction.

Privacy usually implies anonymity, yet the absence of face-to-face social mechanisms for establishing trust exacerbates the difficulty in offering secure and trusted payment systems. And because electronic commerce transactions are several orders of magnitude faster than in the physical world, if it is easy to cheat someone, even for a very small amount, it is easy to do it many, many times over. The costs can add up quickly.

Dan discussed the goals of the Financial Services Technology Consortium (FSTC), which identifies applications of electronic commerce and guides standards development as the marketplace evolves from a paper-based market to a more electronic-based market. The FSTC is interested in modular standards and open APIs to enable interoperation. Standards development is guided by the principle that new systems should be at least as convenient as current systems, such as ATMs. Electronic currency should be similar in style to an electronic check but provide a wider set of payment options, such as direct funds transfer and payment on credit.

The FSTC established the Electronic Clearinghouse, a system of electronic checks and imaging of paper-based checks. The electronic check system is interesting because there is no need for a transaction-time third party (the bank), yet it is just like a physical check transaction. Plans are to extend the system to cover digital cash, debit cards, credit cards, funds transfer, traveler's checks, barter, *etc.*

The imaging of physical checks allows for faster processing and lower costs. Point-of-sale check scanning combined with online transaction-time verification of the account and the amount of the transaction lets the customer tear up the check at the point-of-sale: once it has been scanned, it is never needed again. Fraud prevention and detection schemes need more investigation. Old techniques, such as PINs, may not be adequate in the electronic market because they are hard for users to remember and easy for computers to break. Other techniques, such as neural nets or biometrics (iris scanning, fingerprinting, *etc.*), should be researched. Dan concluded that electronic commerce will likely make life different but more interesting as well.

Jacob Levy opened the discussion by questioning whether humans will trust a system built on intangibles, especially if it includes biometrics. Robert Gezelter expressed concerns about legal ramifications. For example, what will stand up in court? What if a check image from an imaging system was unclear? Is the user liable? To what extent are Uniform Commercial Code issues being pushed down to the level of consumers? Dan emphasized that many of these questions are current research issues but noted that the consumer's world is already full of risks, such as bad checks or counterfeit cash.

Richard Field responded that consumers and commercial enterprises are in different legal arenas and that the end user is often protected in ways that the commercial enterprise is not. But new proposals are eroding the protections of the consumer. For example, it has been proposed that the liability of the consumer on goods purchased with a stolen credit card be raised from $50 to $500. And scarier, a Utah digital signatures act makes the user liable for credit fraud if digital signatures are used negligently. The problem is that we are applying top-level legal concepts to the consumer.

Lee Parks also noted that the general trend seems to be pushing the risk of fraud onto the consumer. For example, many banks now state that if you have not objected to your monthly statement within 30 days of its issue, all records stand as printed. Many commented on the unacceptability of this trend in the customer community. Dan agrees that there are "a lot of minefields."

Cliff Neuman commented that people are willing to give their credit card number over the phone to make purchases, so perhaps perfect security is not necessary.

1.2. Economy and legal

Chaired by Alan Nemeth.

Token and Notational Money in Electronic Commerce
L. Jean Camp, Marvin Sirbu, J.D. Tygar
Carnegie Mellon University

Jean Camp presented the first formal paper of the workshop. Her thesis is that electronic commerce necessitates new legal and social protocols for handling and processing money. New types of currency pose threats that are not addressed in current law. Jean provides a useful and comprehensive collection of case studies examining the relationship between buyer and seller, as well as the requirements and states of knowledge of law enforcement, banks, and observers, for the various types of electronic currency.

A few questions arose regarding nomenclature, *e.g.*, whether NetBill is goods-atomic. There was also some discussion about the best way to view paper transactions. Greg Rose suggested that the work be extended to address international jurisdictions. Eric Hughes surmised that legislation to prohibit new types of money laundering is inevitable.

Economic Mechanism Design for Computerized Agents
Hal R. Varian
University of Michigan

Hal Varian described issues surrounding economic mechanism design, focusing on auction techniques used in practice and their applicability to electronic commerce. In the discussion, Doug Tygar suggested clever ways to "hack" the auction mechanism. Hal offered a "mechanism, not policy" defense. Jacob Levy explored anonymity in auction mechanisms. Hal responded that the value of anonymity would be reflected in price differences.

Can the Conventional Models Apply? The Microeconomics of the Information Revolution
Bruce Don and David Frelinger
RAND Corp.

Bruce Don argued that the information technology industry is invalidating basic legal and regulatory policies and that there is not yet a clear paradigm that regards information as an economic good.

In the discussion that followed, Hal Varian suggested that we don't need new economic theories, we just need to read economics texts from the back to the front — the good stuff is all at the end.

Marvin Sirbu characterized negligible production cost as the key to the economics of software. Dan Schutzer postulated a world of totally free information in which value arises from skillful information management.

Panel Session
Jean Camp, Bruce Don, Hal Varian, Jill Ellsworth

Following brief remarks by Jill Ellsworth, Ph.D., Alan Nemeth led a panel discussion among the authors. Jill is an author, educator, and consultant working with merchants, who are focused more on interoperability than on security. Consumers want systems that are easy, fast, and understandable. Merchants want the system to *work* and they want to get *paid.*

Dan Geer noted a decline in the use of cash, *e.g.*, in grocery stores, replaced by credit or debit cards. "Do not underestimate the lure of convenience." But how can consumers be protected from unwitting violations of their own interests, *e.g.*, their privacy? Hal Varian feels that they can, with a cost. Bruce Don suggested that we may have the opportunity to design the market mechanisms, which ordinarily form of their own accord. This may allow tailoring specific properties of the market, such as those that relate to privacy.

Richard Field described privacy as a commodity to be purchased. Hal Varian characterized it as information of the form "What does he know?"

Ittai Hershman questioned whether consumer issues are as important as enterprise issues. Jill noted that business-to-business commerce is well under way, even on the Internet.

Alan Nemeth opened Pandora's box by suggesting a parallel to the battle between FAX and Email over the last 15 years. More heat than light was shed in response. Much more.

1.3. Payment experience

Chaired by Steve Dusse.

Internet Information Commerce: The First Virtual Approach
Darren New
First Virtual Holdings, Inc.

First Virtual Holdings (FVH), which has been online since October 15, 1994, is in the business of information commerce, not goods and services. Darren New described the problems in supporting electronic commerce on the Internet. The biggest problem is that the Internet is *big.* He also described quite a few ways in which information commerce differs from commerce in real goods. He then described the FVH techniques in some detail.

FVH shies away from cryptographic protocols, which present legal problems as well as potentially interfering with interoperability and usability. In fact, the FVH protocols send no financial information over the net.

In answer to Robert Simoncic's question, Darren revealed that FVH has over 10,000 customer accounts. Darren did not indicate the amount of foreign traffic or how many users employ non-IP protocols, such as UUCP. In the early days, most of FVH's business was done through FTP and SMTP; now it's done mostly via the Web. In response to Marvin Sirbu's question about microtransactions, Darren indicated that sellers are allowed to accumulate transactions before forwarding billing requests to FVH.

Ben Wright asked about the trust relationship between sellers/customers and FVH. Darren replied that FVH is relying on establishing and enhancing its reputation through presence on the net, 800 numbers, relationships with large vendors, *etc.* Dave Crocker made a strong pitch for reputation-based systems.

Arthur Keller asked how FVH handles "varying content." This seemed to be a leading question; Doug Tygar was more direct and asked what fraction of goods sold are erotic. Darren replied that FVH does not study the content of the goods it sells, it just collects the money.

In answer to Andy Lowry's question, Darren suggested that potential sellers of hard goods, such as pizza, are routinely turned away.

Eric Hughes mentioned the potential for Domain Name System hacking to subvert the FVH authorization protocol. Darren confirmed that the potential is there and further that it has been observed in practice, whereupon FVH contacts the responsible authority (*i.e.*, the hostmaster for the domain). This solution may not scale.

Payment Switches for Open Networks
D. Gifford, L. Stewart, A. Payne, and G. W. Treese
Open Market, Inc.

Win Treese described the Open Market system for supporting electronic transactions. A typical transaction involves a Web server with digital offers on it. The customer browses the Web, selects her goods, then presents herself to the payment system to get a digital receipt, which acts as a ticket for goods. The ticket is given to the seller, who conveys the goods.

Open Market supports various payment methods, including credit cards and purchase orders. Authentication can be user-defined, including challenges such as "birth date" or "dog's name." Despite the desire for total anonymity, payment is linked to the good being purchased.

Customer service ordinarily costs several times more than fraud, hence it must be automated. To this end, customers have access to personal, online, up-to-the-minute statements, eliminating the usual monthly statement. Items on the statement contain embedded URLs to more detailed descriptions of transactions.

Because Open Market transactions are inherently nonatomic, good failure semantics are essential. For example, when a customer clicks a button repeatedly, it may or may not mean she wants to purchase several copies of the item.

Answering Hal Varian, Win indicated that Open Market does not yet support discounts based on volume or affiliation. Mark Seiden asked whether receipts are transferable or subject to theft. Win stated that tickets contain IP addresses, among other things, and claimed that tickets are hard to steal, which was met by a general murmur of dissatisfaction.

Jacob Levy observed that in contrast to FVH, Open Market payments take place after goods are delivered. Win noted that FVH itself can be used as a payment system.

John Gilmore complained that he hates shopping, and would like a system that gets him the stuff he needs without his involvement. Win replied that authentication dictates some level of his attention.

1.4. Payment mechanisms

Moderated by Mack Hicks.

Dan Eldridge
DigiCash

Steve Crocker
CyberCash

Dan Eldridge offered some brief remarks. Cash is anonymous and private. Privacy is good. Privacy, once violated, can never be restored. Electronic cash (or coins) is a "bearer instrument" in that the value of the item goes with the bearer. When cash trades hands, the value of the cash immediately belongs to the recipient, in contrast to handing someone a check or performing a credit card transaction, wherein the value goes to the recipient at some unspecified future time.

Steve Crocker followed with some remarks of his own. CyberCash is also building payment systems for the Internet. Their goal is to produce component technology to be integrated into other software products. Their software attaches to browsers and servers. The payment mechanism is very similar to other current schemes. They also provide a peer-to-peer payment structure that does not require the transaction-time involvement of banks.

CyberCash uses cryptography heavily; credit card information goes over the net encrypted. Merchants can neither see nor tamper with customer information but are responsible for forwarding it to the bank.

A CyberCash transaction begins when a server makes an offer to the consumer. The consumer sends back credit card information (encrypted). The merchant sends a message to CyberCash to authenticate the customer. CyberCash obtains a credit card transaction from the bank. (Question from the crowd: "Does it count as a 'card present' transaction?" Answer: No.) CyberCash sends a "yes/no" message to the merchant, based on authorization from the bank. The merchant sends a receipt to customer.

Steve emphasized the importance of a safe, efficient, and fast mechanism, one that could support small transactions. He stated that there are five times as many transactions under US$10 than over US$10.

Mack Hicks initiated the panel session by suggesting that payment protocols are less important to banks than whether certificates such as X.509 or ANSI X.9 are accepted by other banks and payment companies. Steve pointed out that while certificate standardization codifies the binding between a public key and its certification, banks really want to know who you are and whether you will pay your bills.

Richard Field observed that certification authorities

have a tricky balancing act: they must limit their risk without losing public trust. Mack suggested that certification authority may be governmental, reiterating that the process of certificate acceptance is central.

Andy Lowry agreed with Mack and asked Win about liability if a merchant key is cracked, because this has the potential for enormous losses. Win answered that while keys can be invalidated, perhaps based on checking usage patterns, this does not solve the liability problem. Dan said that liability belongs with the key owner; *e.g.*, if a bank's secret DigiCash key is compromised, the bank is liable for proximate losses. Darren echoed this view, citing recent Utah legislation that places liability with the owner of a digital signature. The crowd responded that 1) this legislation has not yet been tested; and 2) this is a good reason to get out of Utah.

1.5. Payment protocols

Chaired by Clifford Neuman.

NetBill Security and Transaction Protocol
Benjamin Cox, J.D. Tygar, Marvin Sirbu
Carnegie Mellon University

Benjamin Cox described the NetBill protocol, designed for server-based electronic commerce of low-cost electronic items, such as those that would be purchased from a digital library. (The initial deployment sites for NetBill are University digital libraries.) NetBill follows a credit-card model and is optimized for micro-transactions and network delivery of goods. NetBill is partnering with VISA and Mellon Bank.

The NetBill protocol is designed so that authorization, payment, and goods transfer are efficient, which will reduce server load, support high volume and availability, and reduce the number of disputes. The goal is for the transaction cost itself to be negligible, to provide for zero-cost goods.

NetBill transactions provide atomicity, non-repudiability, access control, and mutual anonymity and privacy to protect sellers against buyers and buyers against sellers. Transaction atomicity is enforced by a trusted third-party, namely NetBill. Digital signatures are used for non-repudiability. Kerberos tickets are used by customers and merchants for authentication and access control. NetBill runs the Kerberos server.

Tickets have a non-negligible lifetime, which hampers security but improves performance. Public keys are needed for digital signatures, so NetBill extends Kerberos to use public keys for initial authentication. Ben claims this may remove the need for a single, realm-wide Kerberos server.

A NetBill transaction takes place in three phases: price negotiation, goods delivery, and payment. The customer interacts directly with the merchant. A seller can dynamically set the price based on the identity of the buyer, offering discounts based on demographic criteria, such as "the buyer is a senior citizen," *etc.* The customer is also allowed to issue a "bid" message to the merchant.

After the merchant and customer agree on a price, the merchant encrypts the goods, sends the encrypted goods to the customer, and waits for payment. The key is sent to the customer upon payment. A copy of the transaction receipt goes to NetBill — the receipt contains a copy of the key — in case the merchant refuses to send the key to the customer or if the customer misplaces or fails to receive the key.

Both parties digitally sign the payment receipt, using public key cryptography. Certificates encrypted by private key replace the Kerberos ticket-granting-ticket. This is just as good as a ticket for identifying the customer, as it uses the public key to authenticate.

In the question period following Ben's talk, Dave Crocker suggested that the credential and authorization mechanisms used in NetBill are a useful generalization and Ben an his colleagues should consider writing an RFC.

In response to Mack Hicks' question about the origin and form of certificates, Ben stated that NetBill generates the certificates. Their final form depends on the outcome of negotiations with Public Key Partners, Inc. Mack suggested X.509v3.

Mack also asked about liabilities in the NetBill system. Marvin Sirbu replied that an issuer or payment processor must take liability, *i.e.*, NetBill is liable. Doug Tygar elaborated that issues of liability and revocation are clarified in payment protocols in which transactions run through a trusted server.

iKP — A Family of Secure Electronic Payment Protocols
M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, M. Waidner
IBM Watson Research Laboratory and IBM Zurich Research Laboratory

Hugo Krawczyk described iKP, a family of secure multiparty payment protocols based on RSA-1024 cryptography. Modeled on the credit card system, iKP has buyers, sellers, issuers, acquirers, and a clearing house. iKP also supports the notion of a gateway, which translates electronic payment requests into clearing house and authorization tasks.

The iKP family consists of 1KP, 2KP, and 3KP. In 1KP, only the gateway possesses a public and private key pair. 2KP provides merchants with keys, 3KP offers full multiparty security by issuing public/private key pairs to customers as well. The choice of protocol

is based on the security requirements of the application. In general, the protocol's security guarantee is strengthened by requiring that more parties in the transaction possess and use public key pairs. A family of protocols such as iKP allows for gradual deployment as infrastructure requirements are met.

A Set of Protocols for Micropayments in Distributed Systems
Lei Tang
Graduate School of Industrial Administration
Carnegie Mellon University

Lei Tang believes that Internet electronic commerce traffic will require micropayments (US$1 or less), and has developed several protocols to support microtransactions. He uses a simple accounting structure and cheap encryption techniques to keep transaction overhead down.

Lei compared the transaction cost of security requirements, symmetric and asymmetric cryptosystems, and payment models. He concluded that security is necessary to keep transaction cost low, that computational intensity and patent licenses make public keys too expensive, and that a debit payment model is simplest. He then presented three protocols that implement these requirements.

Following Lei's talk, Doug Tygar argued that Lei's protocols are neither goods-atomic nor money-atomic, and that a failure could result in an inconsistent state, such as the disappearance of money. Lei responded that modifications in the protocol could extend them to meet that requirement.

Design Considerations for Lightweight Electronic Payment Mechanisms
Mark Manasse
DEC Systems Research Center

Mark began with some back-of-the-envelope computations. There are 31.5 million seconds in a year, so a computer that costs US$150K per year to run necessitates a revenue stream of one-half a cent per second (assuming constant traffic). If a financial agent skims 2% of the revenue, the flow must be 25 cents per second for the agent to make money. Assuming bursts come in a 10:1 ratio, this requirement is actually US$2.50/second. Assuming the computer can perform 10 encryptions per second, the minimum granularity of a transaction is thus US$0.25.

Improving the granularity of transactions to one cent requires the ability to support 250 transactions per second. To do better than this, say one-tenth cent granularity, requires lumping transactions together, reducing the cost of cryptography, or relaxing some of the guarantees, such as non-refutability, anonymity, or reliability.

Mark then presented Millicent, which uses scrip to support "femtopayments": very lightweight transactions whose costs are fractions of pennies. Millicent is designed for transactions such as a payment per Web access, per stock quote, or per index query. Millicent sacrifices anonymity and non-repudiation to accomplish its ends.

Millicent is a broker that issues unforgeable scrip in large amounts, say US$5, but broken down into small units, each with a unique identifier. Scrip is issued for use with a specific merchant. The merchant verifies that the customer spends each identifier no more than once. Scrip that expires can be refreshed by the broker.

The question and answer period for Mark's talk was lively as participants tried to grasp the details of Mark's system and compare the four systems discussed. Jill Ellsworth asked Mark if concern about government regulation is important in micropayment systems. Mark responded that he has no definite information on the government's interest in micropayments. There was a flurry of responses (most from Marvin Sirbu) that the government is worried about the broad use of encryption, but not its narrow application to electronic commerce. The current status is that the transfer of bulk encrypted documents is legal, but the export of encryption technology is not.

Win Treese wanted to know how a 5% sales tax would be handled on a 1/100th cent purchase. The consensus was to round down.

Doug Tygar suggested that it is difficult to differentiate between so many protocols in one session, especially ones that seem to occupy very different regions of Jean Camp's taxonomy. Cliff Neuman felt that we will have to wait and see how the various systems behave in large volume. Doug added that even though STT appears inappropriate for microtransactions, it should also be considered, because with it, VISA and Microsoft will control a large chunk of the market.

Robert Gezelter asked how a merchant can tell whether scrip is legitimate and how payment authorization is accomplished. Mark responded that the scrip is merchant-specific and that transactions are pre-authorized. Robert questioned whether current laws permit issuance of private scrip. Richard Fields suggested that it is allowed.

1.6. Dinner speaker

Marty Tenenbaum
Enterprise Integration Technologies

Marty Tenenbaum opened his after-dinner address with the observation: "It's been a hell of a year for the Internet ... and you ain't seen nuthin' yet!" A year ago,

when CommerceNet started, people said Internet commerce was either illegal or immoral. Now, according to the Economist, it's bigger than the telephone and approaches the printing press in its impact to society. The Web has unleashed the Internet marketplace, and with the new influx of security into the Web, we'll really see commerce sprout.

CommerceNet was born in February, 1993 at the Sheraton New York (coincidentally, the workshop meeting place), during a meeting of the Technology Reinvestment Program. It is a consortium of over 120 American companies, such as Citibank, IBM, MCI, Netscape, UC Berkeley, and Wells Fargo. CommerceNet is growing at the rate of about 10 new companies per month, with basically no marketing. A CommerceNet Japan chapter is also starting. CommerceNet has working groups exploring, developing, and exploiting emerging technologies. Most of the action takes place in these working groups.

CommerceNet provides a brokering function by disseminating multimedia catalogs; hypermedia browser starter kits (a service no longer felt to be necessary); ISDN connectivity and networking hardware and software; specialized directories, *i.e.*, those not obviated by Yahoo and Lycos and their ilk; security (encryption and authentication); and payment services (credit cards, debit cards, checks). All this works to support collaborative engineering products.

CommerceNet hopes to replace EDI, which requires prior arrangements between trading partners, with Internet storefronts based on Web browsers. Marty described PartNet, a place where parts are available, and the Internet Shopping Network, which replaces rooms full of "operators standing by" with Web servers.

The impact of electronic commerce in the short term will be to cut costs, shrink development cycles, and streamline procurement. The long-term outlook is the atomization, or dis-integration, of vertical companies into ones offering core competencies in niche areas, outsourcing the rest. This will empower small businesses, niche publishing, and tiny markets and will lead to new information services, such as risk management and brokers to bring buyers and sellers together.

2. Wednesday, July 13, 1995

2.1. Technology

Moderated by Peter Honeyman.

Smart Catalogs and Virtual Catalogs
Arthur M. Keller
Stanford University

Arthur Keller opened by asserting that just as important as transaction security and payment for goods is the ability to *locate* things on the net.

Current online catalogs use the Web, in which documents are typically menus pointing to other documents. Each vendor maintains its own menus, an approach that has limitations: it's hard to find what you want when the menus are organized by company as opposed to by product For example, it is not possible to query `www.printers.com` for cross-listing of references to every commercially available printer. Furthermore, there are no standards for organizing information, so each catalog has its own unique structure. In addition, there is often no facility for searching by content and no support for cross-catalog searching.

Some of the queries we might want to issue include: "Give me list of local dealers for the HP DeskJet 1200C PostScript printer"; "Give me a list of all HP printers for the Macintosh"; "Notify me whenever someone announces a color PostScript printer for the Macintosh less than $3000"; or "Give me a list of *any* merchant's color PS printer under $3000."

Support for these types of transactions requires distributed searches; support for "reverse" or content-based search, *e.g.*, to look for printers or disk drives, not product names; support for heterogeneous information sources; getting the information from the horse's mouth, *e.g.*, get information on Hewlett-Packard printers from HP, not from someone else; and cross-searching of multiple catalogs.

The CommerceNet Smart Catalog project supports all of these requirements. The system is composed of distributed agents that speak a common language and translate between it and the individual languages of the merchant catalogs (such as SQL, *etc*.).

The system also allows distributors to make "virtual catalogs" based on smart catalogs. By pointing its virtual catalog at the smart catalogs of its manufacturers, a distributor can provide a consistent interface to its customers. The interface can also be more complete, as it avoids handing off control to the manufacturer, which presents a problem if information must be transferred back to the distributor's catalog service.

CommerceNet queries to manufacturers result in

information in forms like product description = <text>; product picture = <gif>; product specs = <text>; *etc.* An intermediate browser, virtual catalog, or CAD tool can do with the information as it pleases, allowing distributors to customize and personalize their Web page layouts and organizations.

The system also contains Facilitators, or intermediary agents along the lines of X.500 directory agents, responsible for keeping pointers about specific product types, such as TOASTERS or PRINTERS or FOOD. Finally, Smart Catalogs support Ontologies. There must be standard languages/grammars for each product type available via the system. Standards organizations will define the highest-level languages, and every database must use the defined names for item labels. Lower-level languages, as needed per specific product, can be defined by the manufacturer of the product.

Robert Simoncic asked whether CommerceNet offers a new way to get into price wars. The distributor does not need to stock the product, so what stops each manufacturer from bidding down at transaction time? Arthur speculated that new business and commerce models will appear for precisely this reason.

Alan Nemeth pointed out that caching the results to queries may be a problem if these results don't remain valid for too long. Arthur agreed that this is important and will be looked at when virtual catalogs are built.

Robert Gezelter asked about bottlenecks and system overload. Arthur joked when you know an important bid is coming, you could swamp your competitor's catalogs with queries. In seriousness, he advised that it is possible to put a filter on the system, or send information out only to accepted endpoints, but this is also future work.

Safe Tcl: A Toolbox for Constructing Electronic Markets
Jacob Levy and John Ousterhout
Sun Microsystems Laboratories

Jacob Levy spoke about electronic meeting places and using Safe Tcl to build them. Electronic meeting places are virtual spaces where mobile agents (represented by code+state) can run. The spaces have persistent state and provide a consistent view to all of the mobile agents currently in a room. Possible uses include advertising and commerce, social interactions, integrated currency, advertising methods, ordering and delivery of goods, *etc.*

Safe Tcl offers a way to run these electronic meeting places. Safe Tcl handles multiple address spaces within a single process by running a different Tcl interpreter in each address space. This allows execution of code that is not trusted or of communicating scripts that do not trust one another. System calls and IPC requests are processed by intermediary agents that allow actions based upon authentication.

A Safe Tcl environment has a hierarchical structure in which parent threads create children threads. Each thread is a separate Tcl interpreter. An interpreter's execution restrictions are set by its parent. Children must trust the parents, but the parents do not trust the children.

Safety is provided by ensuring that only known good code is run by unrestricted interpreters. Untrusted code is run by restricted interpreters.

In Safe Tcl's underlying trust structure, trust is not based on *where* code is from, it relies on *who* code is from, authenticated using digital signatures.

Multiple interpreters with different restrictions isolate concurrent executing scripts. Removing functionality makes them safe from each other.

Open problems in safe execution include protecting a mobile application (a script that moves between meeting places) from the hosts, persistent scripts, and the absence of standards.

Ben Cox asked about combining safe elements in unsafe ways, for instance, a pop-up box that asks for the user's credit card number and then relays the information to someone else. Jacob replied that the problem of user gullibility is something that he plans to address.

Arthur Keller asked how Safe Tcl decides what scripts run on which interpreter. For example, you don't want a privileged interpreter to be able to grant person A file access to a script written by person B. Jacob responded that scripts are authenticated as if they are the actual person; a script cannot do things that the person cannot.

Developing and Deploying a Corporate-Wide Digital Signature Capability
Diane E. Coe and Frank J. Smith
The MITRE Corporation

Diane Coe described current cryptographic efforts at MITRE, which include digital signatures, a key management system, distributed authentication, and a system integrating these various aspects. MITRE's interest in digital signatures stems from a belief that they can help lead to the paperless office and thus save time and money.

MITRE's efforts are based on the RSA BSAFE toolkit, using RSA for authentication and MD5 for hashing. SIGN and VERIFY functions are part of a software package that is easily embedded into other applications.

MITRE uses a key management system integrated with the X.500 directory structure for authentication. Users go to a key generator for a key. The generator places the key and a certificate onto a floppy disk or

smartcard. The certificate is also stored in a certificate database and in the user's X.500 entry.

MITRE uses Kerberos-based distributed authentication. This provides single sign-on capability using Kerberos credentials and passwords for proprietary and residual systems. They are moving from floppy disk to smartcards for security reasons.

MITRE has run into horrible interoperability problems: only rarely can credentials be shared between different systems. Most products are not open and the ISO smartcard standard is still evolving, so interoperable products are only just emerging.

MITRE's Kerberos realm ends at the edge of MITRE. To communicate with the rest of the world, MITRE has followed the relevant standards: X.500, X.509, Kerberos, RSA, *etc.*, but there is not yet any advantage to having done so. Credit card companies are building their own systems and digital signature capabilities; MITRE fears it will need a different smartcard for each bank. "Everyone says they're singing from the same sheet of music, but they're not singing the same song."

Marvin Sirbu asked whether the fact that a token contains only one key-certificate pair means that you need a smartcard for every certificate authority. Diane responded that this depends, *e.g.*, on whether VISA uses your key or issues their own. Marvin suggested that if you can set up the token so it can hold multiple certificate-key pairs, then you solve the problem. Greg Rose suggested that you get a certificate-key pair out of each token and put them into a universal one.

Greg asked Diane why MITRE didn't go to a hierarchical system instead of one certificate authority. The answer was simple: Cost.

In response to Juan Rodriguez' question, Diane confirmed that the date on the floppy signature is protected by user password encryption.

Answering Arthur Keller, Diane said it is impossible to masquerade as someone if you find their floppy or smartcard, as you also need their PIN/password to unlock the smartcard.

John Gilmore wondered why MITRE chose the ISO smartcard over PCMCIA, as the latter offers access to many more readers. Diane responded that the smartcards fit into MITRE's existing badge readers.

Bob Gezelter asked whether the credentials in the smartcard go over the network. Diane agreed that would be a huge security hole, and stated that the credentials are read by the card reader but go no further than that in the clear.

2.2. Extensions

Chaired by Marc Donner.

Digital Notary
Stuart Haber
Bellcore

Stuart Haber described Surety Technologies, a commercial spin-off from Bellcore whose initial product is a digital notary, used to prove that a given document existed in a certain form on a certain date. A sample use is notarized electronic lab notebooks to defend a patent.

Stuart provided a concise overview of the operation of the notary. Customers convey digital documents to Surety. Periodically, Surety constructs a binary tree, with the MD5 hash of each document stored at the leaves. The interior nodes are also labeled with hash values constructed from the combined values of the node's immediate children. The value of the node at the root of the tree is published (in the Sunday New York Times). Each customer is then given the hash value of her leaf document, as well as the hash values of all nodes adjacent to the nodes on the path from root to (customer) leaf. With these hash values in hand, the customer can prove that her document contributed to the published hash value.

The hash function is one-way, so possession of a collection of values that produce the published hash value is firm evidence that the customer's document existed prior to the publication date. Furthermore, because the hash values are constructed in a tree, the number of values returned to the customer is equal to the height of the tree, so the procedure scales with the log of the number of documents. Additional scaling can be had by allowing local servers to build their own trees, transmitting the root hash value to a Surety tree. The system has no keys to compromise and appears to be extremely secure.

In answer to Richard Field, Stuart reiterated that Surety need not retain copies of the documents or even the intermediate or root hash values. The customer is given all the tools and values necessary to recompute the root hash value.

Generic Extensions of WWW Browsers
Ralf Hauser and Michael Steiner
IBM Zurich Research Laboratory

Ralf Hauser described the need for a "snap-in" product to put into Web browsers that offers security and payment methods. He found that this is not really possible: you need to modify servers or add processes to systems. Ralf went on to describe a better way to architect Web browsers, but the audience resisted the notion of changing everyone's browser or even considering

more work in this area when Java and Safe-Tcl are just now appearing.

The initial goals of generic extensions to WWW were to support iKP security without modifying browsers. Current needs include security when making purchases. Several possible architectures were described. One relies on MIME to invoke a local payment client, which contacts a payment server to complete the actual purchase. This does not easily admit post-processing. Another architecture uses a proxy Web server that intercepts and processes payment requests; firewalls can make this quite challenging.

Ralf argued in favor of a generalized architecture, which includes an extensions manager, passive extensions such as filter viewers, and active extensions such as remote controllers. The extension manager handles not only MIME, but also HTML as an HTML-viewer extension.

John Gilmore asked how to accomplish these ends in a way that is not specific to any particular browser. Ralf answered that while binary compatibility is required, it's as generic as MIME. Rohit Khare elaborated by comparing to CGI, which differs among the various platforms. But John would not be satisfied with anything less than Java or Tcl.

Rohit asked about versioning for the client-side, to make sure the latest version of security is being used. Ralf suggested placing the version number in the MIME type, as the architecture is transparent to that issue.

Andy Lowry observed that the generic extensions approach doesn't push in a direction for patching together distributed applications, especially long-lived ones and gave the IBM SOM product as an example. Ralf felt that because HTTP is stateless, SOM could be supported under the architecture.

Andy followed up by suggesting the importance of long-lived processes internal to an organization. Ralf responded that the work described here is in the vein of a browser and renderer; long-lived processes can be launched from the browser, but anything more is beyond its purview.

Secure Coprocessors in Electronic Commerce Applications
Bennett Yee and J.D. Tygar
Carnegie Mellon University

Bennett Yee characterized the fundamental problem of electronic commerce as the distribution of security and argued that hardware should play a role. Secure coprocessors are capable of solving some of the problems that software solutions can't.

A secure coprocessor is a standard microchip and some NVRAM in tamper-proof packaging. Potential form factors include chips (such as the infamous Clipper/Capstone), smartcards, or PCMCIA. Bennett maintains that technology for physically secure hardware is a reality.

A secure coprocessor is a safe place to store cryptographic keys as well as to do some computation. In one application, a secure coprocessor can be used to fight software piracy by enabling delivery of encrypted software. The secure coprocessor holds the crypt key and runs the decrypted code. Everything sent to the host machine, including paged data, is encrypted before it goes out. This scheme protects applications, but not data. For example, a protected picture must be decrypted to view it, at which point the protection may be lost.

A secure coprocessor provides a very simple scheme for electronic money: the coprocessor knows how much money it has allocated to it, and tampering results in loss of cash. Such a system can be much safer than DigiCash, where it is possible for a merchant to not receive the packets necessary to reconstruct the money (and here, Bennett gave an extremely effective demonstration of the fragmentation of digital money over the net by ripping up a dollar belonging to Ben Cox), or to pretend that it never got the packets. Using secure coprocessors, neither side of a transaction can be tampered with and protocols can be run as transactions, with ACID properties guaranteeing that money is either transferred completely or not at all.

Point-of-sale is another potential application, using a secure coprocessor as a debit card. To protect against Trojan-horse merchant systems, the coprocessor should allow the customer to enter a password to approve the withdrawal of funds and to verify the amount. Bennett described a simple method for ensuring that the merchant system does not get the password: the coprocessor displays a random string and the customer uses arrow keys on merchant's system to change the values to the correct password.

A secure coprocessor also allows transmission of sensitive materials to other processors using public key encryption. Such a scenario appears to be impossible with Safe Tcl or Java.

Mack Hicks asked whether secure coprocessors can offer virus protection. Bennett answered in the affirmative, and outlined one scheme.

John Gilmore asked how to protect users from malicious cards that do not allow them to "read the code." There seems to be no satisfactory answer, but Bennett favors openness.

Eric Hughes delved into atomicity issues in

communications between coprocessors over unreliable networks; Bennett referred him to the literature on two-phase commit protocols.

The DigiBox: A Self-Protecting Container for Electronic Commerce
O. Sibert, D. Bernstein, and D. Van Wie
Electronic Publishing Resources

Olin Sibert argued that one of the requirements of electronic commerce is a container mechanism that protects itself as well as its contents from tampering. For example, currently an author writes a book, or an artist composes music, *etc.* When it comes to being reimbursed for the actual number of copies produced and sold, the creator of the work is at the mercy of the publisher, with no way to ensure that the number of copies the publisher claims is the number of copies actually out there. This is compounded by consumers making their own copies of the work, cheating the copyright system. How can the rights of the creator be protected?

One of the main disincentives for breaking copyrights is purely physical: the price of a book is generally lower than the hassle to sit at a copier for the many hours it takes to make the copy. The computer and digital information technologies open up new ways to be exploited, but also open up new ways to protect copyrights.

Olin described DigiBox, a generalized container that holds and protects arbitrary information content. Users pay for the keys before the container will hand over contents. Containers can hold many different items, *e.g.*, several books, movies, audio recordings, and divulge only one part at a time.

A complex key management system allows portions of the keys to be divulged, with items in the container being encrypted by several different key fragments.

Olin's scheme uses selective encryption for performance reasons: encrypting a small fraction of an MPEG file makes it just as useless as one that is fully 100% encrypted. For audio, the entire file is encrypted, but the key changes periodically. Each key can be less secure (for faster implementation) but the net effect is no loss of security.

DigiBox has many applications, such as discount coupons, key escrow, arbitrary content, *etc.*

Noting that the key mechanism is extremely complex, Marc Donner asked what sort of attacks it is meant to defend against. Olin responded that DigiBox allows multiple copies of a mass-produced product to have different keys for opening it. Part of the complexity is to ensure that one divulged key can not open every copy of a container.

John Gilmore cautioned that software and keys don't mix very well in the real world. Olin suggested that an approach that combined DigiBox and secure coprocessors might offer more stringent security.

Kerberos Plus RSA for World Wide Web Security
Don Davis
Consultant

Don Davis, veteran security guru from the Kerberos project, described his displeasure with other Web security systems and suggested an approach that combines strange bed-fellows Kerberos and RSA with minor protocol enhancements. Don's gripe with SSL is that it doesn't authenticate servers, just clients, which may leave consumers vulnerable to credit card fraud. SHTTP, on the other hand, makes too many homogeneity assumptions, forcing both sides to use either public keys or Kerberos.

Don's scheme requires some changes to the Kerberos protocols. Touted advantages include obviating the necessity of a key database (thus diskless operation) and improved revocation. The scheme seems to have many administrative and performance advantages.

In an interesting side-comment, Don recently discovered and documented an oversight in the Kerberos protocols: it turns out that Kerberos client clocks don't need to be synchronized.

Bennett Yee asked about Trojan horse attacks in distributing patches; the crowd responded "CERT." Don pointed out that distributing patches is an iffy way at best to correct existing ills, as there is no certainty that vulnerable sites will install patches, in fact bitter experience proves otherwise.

2.3. Luncheon speaker

Alliance Ecology: A Key to Strategic Success in Electronic Commerce
Bud Mathiasel
Ernst & Young

Bud Mathiasel, Director of Multimedia Consulting Services for Ernst & Young, opened by asking the audience who will benefit from a global multimedia market? The answers included Microsoft, Sony, Netscape, Disney, SGI, AT&T — all the usual suspects. Bud maintains that no one will do it alone, and offered many examples of the alliances being formed among information technology players, as everyone realizes the risks in going it alone.

As bandwidth grows, so does the complexity of its use as applications take advantage of its potential. Bud identified a general trend, which he called "disintermediation" — the removal of intermediary points between context, content, and infrastructure. (This is similar to Marty Tenenbaum's notion of dis-integration

of vertically integrated manufacturers.) As an example, Bud observed that automated teller machines eliminate tellers, giving customers a more direct association with their banks.

Other trends are the movement from marketplace to "marketspace" and from "clockware" to "swarmware." Peter Honeyman to Andrew Hume: "Kill me now."

In general, Bud's comments left folks reeling. There were many polite, but confused, questions. Jacob Levy was heard to observe, "What he's saying is that we want a kinder, gentler marketplace."

2.4. Electronic commerce in the real world

Moderated by Win Treese.

Darrell Davis
Lombard Technology Group

Andy Lowry
Morgan Stanley

Howard Alt
Sun Microsystems

Andy Lowry began by describing Morgan Stanley, which carries out extremely high-value transactions for a small client base in a highly regulated market. Morgan Stanley uses the Internet for commerce, in part because of "a critical mass of clients and business partners there" and is now exploring the use of the Web as a generic service delivery platform already in place with a clean split between client and server. Using the Web is also good for corporate public relations.

The big question, naturally, is security. Authentication and authorization are absolutely critical. If unauthorized people back out on large transactions, Morgan Stanley is stuck with the costs. The solutions being employed include key escrow, trust management, and risk management.

Open questions abound, such as the liabilities when a trade is based on a certificate whose key has been compromised, or how to place performance guarantees (time of delivery, or even delivery at all) on the net. For example, if someone submits a TRADE message just before the close of the market, and the delivery of the message happens *after* the close, who is responsible? Answering his own question, Andy suggested that insurance, classic risk management, will play a role.

Next, Darrell Davis described some of the challenges faced by Lombard Technology Group, which will soon be selling stocks and other instruments through the Web. Lombard uses the Netscape commerce server through a firewall to provide portfolio management for its clients.

Some of the challenges faced when a naive management decides to go mission-critical with a new and popular service include a boss who is gung-ho about performance but not about security; neglecting customer support in initial plans, requiring developers to be pulled off other work; providers that don't support SSL or even Web browsers; and exponential growth in traffic.

Howard Alt spoke next, explaining that Sun sees a different picture of the world. Sales do not go to the customer: they go through channels. The sales channel creates demand and enables volume. When you bypass your channels, you aggravate them. It is a bad thing to aggravate sales channels.

Sun regards the Internet as a means of delivery, not a channel, and wants their electronic agreements to include sales channels. He also pointed out that while Sun wants copy protection, key-based service denial is not their goal. (It is a problem when a customer has bought a software package but cannot run it because the license server is down.)

Black-box technology is bad: magic secrets are not reasonable in the real world. It is hard to control too many secrets. In contrast, the cryptography people got it right: the algorithms are public knowledge, the implementations are public knowledge, only the key is secret. Limiting the number of mysteries is a worthwhile goal.

In the panel discussion, Darrell reported that Lombard receives about 100,000 HTTP requests per day, about a third of them for stock quotes, and the rest for graphs. 400 requests per day for new account information is typical, and about a quarter of these result in new customers. Lombard's server generates graphs dynamically with Gnuplot, which is killing the system, as is a horribly slow RAID 5 system. Other than that, the server has sufficient capacity.

Darrell would like to have better tools for authentication. Lombard uses account number and password, similar to other trading companies, and not challenge-response because it is hard enough to get clients to plug in their modems and use the right browser.

Rohit Khare asked why we are content with weak security on the phone and yet we expect so much from the Internet. Bob Gezelter replied that it is easy to tap into the net and overhear thousands of circuits, but tapping a phone line taps only one circuit.

Andy observed that we're going to have a very rich set of certificates in the future. Howard noted that while the Web is great for retrieving information, it lacks management tools and standards necessary for commerce. Mark Seiden suggested that stable URLs are a valuable asset, just like a stable 800 number.

2.5. Breakout sessions

In the breakout session, the workshop broke into small discussion groups to address issues of liability, offshore jurisdictions, privacy vs. anonymity, and intermediate success and failure in economic commerce. The workshop then reconvened to discuss these issues in the larger forum. A panel with one representative from each group was formed by Ittai Hershman, Jacob Levy, Peter Honeyman, Dan Geer, and Eric Hughes.

Who is liable?

Eric Hughes' answer was simple: someone else. Ittai Hershman suggested that laws have always been slow to adapt to new technologies; we need to think in terms of the credit card paradigm (which includes a customer, a merchant, and a bank). As far as liability, people will sue the person they think *can* and *will* pay. Joe Arceneaux offered a comparison to hazardous goods, where a broad spectrum of risk-allocation methods are already in place.

Dan Geer offered a new perspective: the person who needs to believe is liable. If I need to trust a fact and it turns out false, I am liable. If the fact is backed by a guarantee, and the guarantor needs to believe something else (like the customer will pay, or the customer has been authenticated), the guarantor is liable. Ben Cox noted the similarity to PGP's "web of trust."

Lee Parks suggested that this is similar to a consumer's responsibility to protect her credit card and PIN number; it is her obligation to report the theft of her card immediately. Lee offered a question from the hard-boiled school: When bad things happen, who will the legal system point the finger at?

Richard Field held that the belief argument is circular: you're liable because you need to believe, and you need to believe because you are liable. Ittai Hershman observed that courts are likely go with precedence rather than shift to new, untested rulings. Eric Hughes made a comparison to warrants in existing systems: a check implies the existence of funds in your account.

Peter Honeyman argued that liability is built into the price of the transaction based on a market for risk acceptance and management.

Privacy is a right; anonymity is a shield for criminals.

Ittai Hershman pointed out that different levels of anonymity exist. When you go down to the corner video store and rent a porno video you have *virtual* anonymity in the implicit assumption that the clerk will not reveal your name even if he recognizes you. If you purchase your own copy for $20 in Times Square, you have *real* anonymity. Ittai suggested that society will not tolerate *real* anonymity on the Internet.

Jacob Levy claimed that privacy is a commodity, for sale like any other. There was widespread disagreement with this position. Andrew Hume argued, "That's a bald-faced lie. You can't buy privacy *today*." There was general dissent to this, as well. Greg Rose suggested that if you can buy privacy, someone else can buy a breach of it.

Joe Arceneaux argued that even if anonymity IS a shield for criminals, it has its uses and good points. Privacy is a policy issue; anonymity is a technology issue. Dan Geer pointed out that anonymity is not binary, *i.e.*, it's not the case that you either have it or you don't. Rather, one has privacy within a certain domain. For example, it is desirable to distinguish between one's private and commercial lives: it is not advantageous to offer someone anonymity while on the job if the company is liable for the worker's actions. Furthermore, while anonymity is the only shield against data aggregation, the law will inevitably require *some* breach of it.

Peter Honeyman offered the view that the central question is one of social control. For example, Swiss bank accounts pay less interest on anonymous accounts. Privacy seems to have less value today than it has had in the past.

Eric Hughes offered four maxims: anonymous systems are hard to audit and adjudicate; anonymity decreases as the transaction size increases; "when cash is suspect, only suspects will have cash"; and liquidity substitutes for identity.

Dan Geer offered an alternative viewpoint: anonymity is the best shield *against* criminals. Josh Rabinowitz agreed that anonymity protects "normal" people.

Andrew Hume maintained that privacy is all the average person *really* wants. Anonymity is nothing more than an implementation issue.

Eric Hughes gave two examples. Chaum's DigiCash provides anonymity to the common man. On the other hand, Morgan Stanley has run full-page ads offering anonymity (actually, pseudonymity) for large financial transactions.

Ben Cox suggested that changing a pseudonym often enough effects anonymity, but this provoked widespread dissent. Eric Hughes pointed out that only the first use of a pseudonym is truly anonymous. Peter Honeyman noted with some sarcasm that "those who have nothing to fear have nothing to hide." Dan Geer held that electronic commerce promotes data aggregation and that anonymity is the only protection.

Andy Lowry asked whether we should be doing something about these issues besides talking about them, such as forming a CommerceNet working group.

What will move offshore?

Joe Arceneaux proposed that in the short term, socially disapproved industries such as pornography and gambling make the transition to offshore operation. A long term is less sure. The big question is privacy: if going offshore increases the level of privacy, it is an enormous advantage. However, many activities will remain traceable, and it will be harder and less desirable to set up your own business offshore. Dan Geer pointed out that today's networks already make things location-independent. The question is what would keep a business *onshore*?

One possibility is the protection of the legal system, which suggests that you would choose a physical location for its legal codes. Electronic services will likely move offshore because there is no disadvantage, such as shipping costs. Industries that respond to bids, like artwork, copy writing, programming, *etc.*, will easily be placed offshore. Also, support operations are likely to make the transition because of reduced costs; for example, today, many 800 number support lines go to places in the Midwest for economic reasons. When you call an airline, or the mail order catalog, the person answering the phone is usually *not* located in the corporate headquarters because of *cost.* These types of operations will go wherever it is most cost-effective.

Peter Honeyman suggested that operations will move offshore because it is economical or required to do so. When people see that the Lithuanian Web server is generally underloaded, there will be a rush to move there. Peter questioned the efficacy of moving out of a jurisdiction, raising the example of the California erotic BBS convicted of violating Memphis community standards.

Eric Hughes replied that Internet carriers are enhanced services providers, *not* common carriers, so that different laws apply to them. Amplifying Eric's comment, Hal Varian noted that a common carrier is protected from liability for the contents of what is carried, while enhanced services providers *are* liable for the contents. Arthur Keller pointed out that the question of common carrier status can be murky, *e.g.*, cable TV providers are liable for the contents of their transmissions.

There was a general feeling that much software development will move offshore, enabled by the Internet. Peter noted that the University of Michigan Digital Library project has its digital scanning done in Barbados.

Eric suggested that while moving offshore has anonymity and regulatory advantages, these are offset in part by a disadvantage in restrictions on import/export.

Who dominates electronic commerce in the year 2000?

There was general agreement that the entertainment industry will be a big winner.

In Dan Geer's breakout group, the primary question was what the infrastructure will look like. Dan restated the question by asking whether the dominant players will be new, unforeseen businesses enabled by electronic commerce or whether they will be revamped, old businesses. Will service providers like AOL, Compuserve, and Prodigy still be around? Will there be a global provider or regional carriers? Another fundamental dichotomy: Which wins more, content or context, *i.e.*, the stuff or the transport?

Peter Honeyman suggested a different dichotomy: Spielberg vs. CU-See-Me (extended to CU-See-Me-CU, *etc.*), *i.e.*, the little guy or the big guy? Clearly, VISA, Mastercard, AMEX will win; all schemes presented involved the credit card companies in one way or another. Peter's one-word summary: turmoil.

Joe Arceneaux agreed that credit card companies will win, as well as other transmogrifications of present financial systems, enabled by the new commerce paradigm.

Jacob Levy offered the view that one big winner is the customer, who will have a much wider selection of choices than ever before. Winners will include the providers of funding, hardware, and information. Losers will include banks that do not adapt and the poor, who will suffer lack of access. This was met by some disagreement over whether the rich and the poor will have such a great degree of disparity in access to electronic services. Jacob asked how many poor people have bank cards. Someone in the crowd asked how many have cars.

Peter suggested that the decreasing cost of technology has a democratizing force. Joe Arceneaux said that when Al Gore spoke in South America about the Internet as a force to bring democracy to them, his audience was less than enthusiastic, as they seem to hold a dim view of American democracy. Richard Field pointed out that third world countries that have missed out in the last few generations of technology can leap-frog cheaply and easily into the global mainstream. Andrew Hume pointed out that in surprisingly affluent places, you still see people who cannot afford even modest monthly connect fees. Peter argued that we're moving the cost line down. Andrew replied that we're moving the poverty line up. Andrew went on to say that when Bell Labs tried to offer free access to public libraries by donating machines, the libraries couldn't afford the cost of continuous access.

Eric Hughes suggested that nobody will dominate, just as no one "dominates" today. Obvious winners include the credit card systems. Obvious losers will include paper mail-order businesses, salespeople and securities traders who lose their jobs, and technical people like Web administrators will lose quality of life.

Andrew Hume argued that the mail-order industry is making *so much money* that electronic commerce would need to *completely* swamp the market to hurt them. You won't hurt L.L. Bean for a *long* time. It was pointed out that L.L. Bean is on the net already.

Ittai Hershman stated that capitalism will be a big winner.

Richard Field felt that the federal government will definitely win.

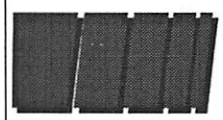
2.6. Wrap-up

Daniel E. Geer, Jr. Sc.D.
Open Market, Inc.

After an enthusiastic show of interest in holding another workshop in the near future, Dan Geer offered some closing remarks and adjourned the workshop.

Acknowledgements

This digest was produced with the assistance of student scribes, Sarah E. Granger, Bruce Jacob, Trent Jaeger, and Charles Thayer, who diligently recorded the presentations and followup discussions; their efforts are warmly appreciated. Scott George lent his quick and careful eye to a draft of this digest. The workshop itself was the brainchild of Daniel E. Geer, Jr., Sc.D., who accomplished the impossible by taking it from conception to fruition in six and one half months.



Electronic Banking and Electronic Commerce on the SuperInformation Highway

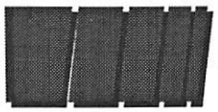
July 11, 1995

Daniel Schutzer, VP & Director of Advanced Technology, Citibank

President, FSTC

What is the Information Infrastructure?

- Home Shopping, Movies on Demand & Set-top box
- On-line services; e.g. Prodigy, AOL, Compuserve
- Telephone, Cellular and Data Networks
- VAN's & EDI
- Internet
- Electronic POS
- All of the Above

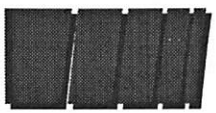


Outline

- What is the Information Infrastructure?
- What's all the excitement about?
- What's Electronic Commerce?
- Financial Services: both enabler and user
- Needed Infrastructure and standards
- Role of Consortiums

What's All The Excitement About?

- A revolution in how we work and play
- Convergence of industries and technologies
- Fundamental alteration/realignment of traditional roles and value-chains
- Both a threat and a promise
- Lots of uncertainty, chaos and rapid change



Internet Statistics

- Over 35 million users and growing 1 million/month
- International in scope
- Already the size of a country
- Is heavily used and not just by technocrats

Electronic Commerce

- Basic activities:
- Shopping/Advertising
- Negotiating
- Ordering & Billing
- Payment & Settlement
- Accounting
- Ancilliary Services (e.g. loans, bonding, escrow)

Advantages to consumers and businesses

- **Advantages to Consumers**
  - **Convenience**
  - **Faster service**
  - **One-stop shopping**
  - **More personal control**
- **Advantages to business**
  - **Financial Agility**
  - **Faster service**
  - **Reduced costs for infrastructure & reduced risk**
  - **More efficient cash flow management**
  - **Opportunity for reaching new markets & customer**

Financial Services as an enabler

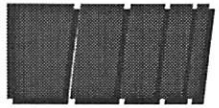
- Both an enabler and a user
- To conduct commerce over the infrastructure we need:
  - > **Secure & trusted payment & credit mechanisms**

Financial Services as a User

- The virtual bank
- No physical boundaries
- 24x7, anywhere, anyhow
- Accessibility to people and systems
- Personalized, full function service
- Secure, convenient & transparent
- Global scheduling

Distinguishing Attributes of NII

- Not face-to-face
- Information in form easily modified by computer
- Huge increase in rate of change/speed & volumes
- Increased difficulty in authentication & authorization



Key Needs

- Privacy
- Intellectual Property
- Authentication
- Authorization
- Non-reputability - "Signing"
- Secure, reliable payment
- Isolation/Protection of Corporate assets/data

Current Payment Mechanisms

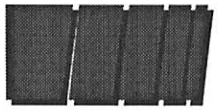
- There is a rich variety to meet individual preferences:
- cash
- checks
- charge cards
- debit cards
- travellers checks
- Other: bank notes, letters of credit, barter

Proposed Electronic Payment Taxonomy

- Locally stored tokens vs. On-line server
- Registered/ID
- Auditable/accountable
- Authentication - what/who
- Links to owner & counterparties
- Degree of anonymity
- Actual value or proxy
- Links to accounting systems
- New concepts - meterware/micropayments

Cluttered Landscape in 1995

- Netscape, First Data ,Bank of America, Mastercard
- Cybercash
- Open Market
- NetBill
- DigiCash
- First Virtual & EDS
- Microsoft & Visa
- SmartCash
- Mondex
- AT&T
- MCInet
- NetCash & NetCheque
- American Express and AOL



Some Alternative Strategies?

- **Proprietary closed - vertical stove-pipe**
  - » **e.g. EDI VAN**
- **Standards-based**
  - **– Ad-hoc Standards**
    - » **Proprietary defacto - horizontal; e.g. IBM-compatible PC**
    - » **Open; e.g. Internet**
  - **– Formal**
    - » **e.g. ANSI**

Different Ways to Achieve Interoperability

- Open Systems
- Gateways
- Multiple Homing
- Drive everyone else out or relegate them to such a small market as not to care
- Open seamless interoperability among applications is hard - because constant migration between infrastructure & applications

Conventional Wisdom

- Proprietary closed is most desirable:
- Can control knowledge of product
- Can keep customer base captive
- Can demand higher profit margins
- Can be quicker reacting with new products & releases
- Can create barriers for competition

On the other hand....

- If proprietary, but ad hoc standard
- Others can add value to your product
- Don't need to be able to dominate to same extent
- Good, perhaps in some ways preferred strategy

Additionally.....

- **Users prefer open - more choices, better price-performance**
- **If low enough cost barriers - new customers and new uses become possible**

  →its about changes in social behavior/social experiment
- **With network services, the more users the greater the value**

  →positive feedback
- **Can succeed in open environment through agility, innovation and "networked alliances"**

Examples & Trends

- Silicon Valley
- Increase in on-line businesses with commercialization of Internet
- Including more business for Prodigy, Compuserve, AOL
- Creation of new business, new opportunities
- Trend appears to be for things to be increasing open - moving up layers, with increasing modularity/granularity

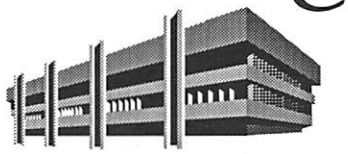
Need for Infrastructure

- Rapid changes in technology creating new challenges
- Need to make services more accessible
- Need for standardized electronic and application programming interface
- Increased threats to security & privacy
- Increased costs of operational problems
- Need to break walls between vertical stovepipe solutions

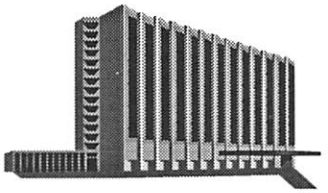
Value of a Consortium

- Early information about emerging technologies
- Facilitates rapid development & introduction of standards
- Effective voice in shaping and directing change
- Spreads cost & risk & fosters cooperation
- Protects intellectual property
- Brings multiple viewpoints, external ideas

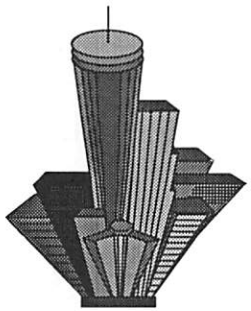
Financial Services Technology Consortium



Universities

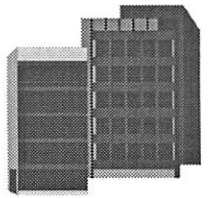


U.S. Banks

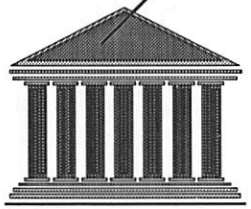


U.S. Banks

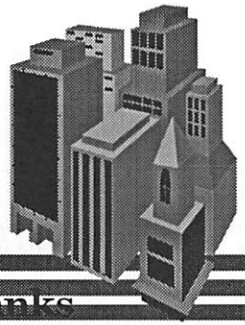
F S T C



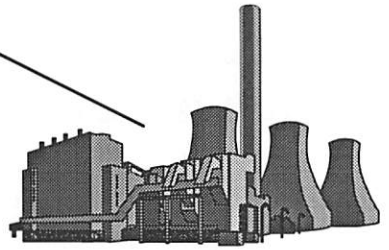
U.S. Banks



.U S Treasury



U.S. Banks



D.O.E. National Labs

Mission and Charter

The FSTC is a consortium of financial services providers, national laboratories, universities, industry partners, and government agencies whose goal is to enhance the competitiveness of the United States financial services industry.

FSTC sponsors non-competitive collaborative research and development on interbank technical projects affecting the entire financial services industry - with particular emphasis on projects involving the development of the National Information Infrastructure (NII) and High Performance Computing and Communications Initiative (HPCCI).

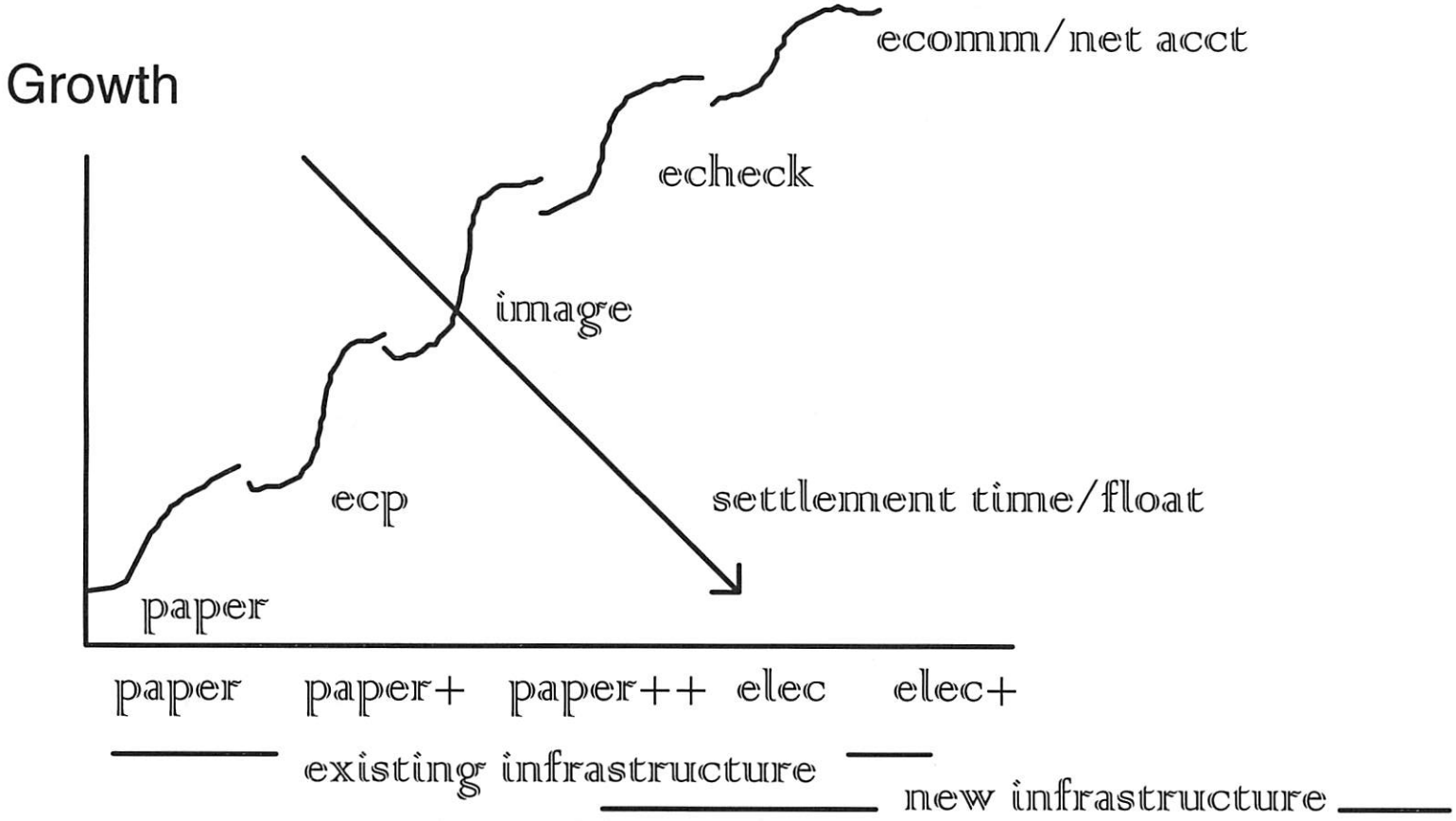
Financial Services Technology Consortium

- **Current members:ACH, American Express, Bank of America, Bank of Boston, BancOne, Bank of Montreal, Barnet Banks, Cardinal BancShares, Chase Manhattan Bank, Chemical Bank, CoreStates, Citibank, Broadway & Seymour, Equifax Credit Information Services, ECCHO, Huntington Bancshares, MasterCard, NationsBank, VISA, Wells Fargo, NYCH, U. S. Post Office, Lawrence Livermore, Los Alamos, Oak Ridge and Sandia National Laboratories, Bellcore, National Security Agency, Columbia University, Polytechnic University, Stanford, UC Berkeley, AMS, AT&T, BBN, CU Cooperative Systems, Cybercash, DEC, Heuristics, IBM, Lifecycle Technology, Motorola, Oki, National SemiConductor, Proprietary Financial Products, RDM, Sun Microsystems, Tower Group, Unisys....**

Major FSTC projects

- Electronic Commerce
- Electronic Check
- Inter-bank Check Imaging
- Fraud

Relation of Projects to Vision



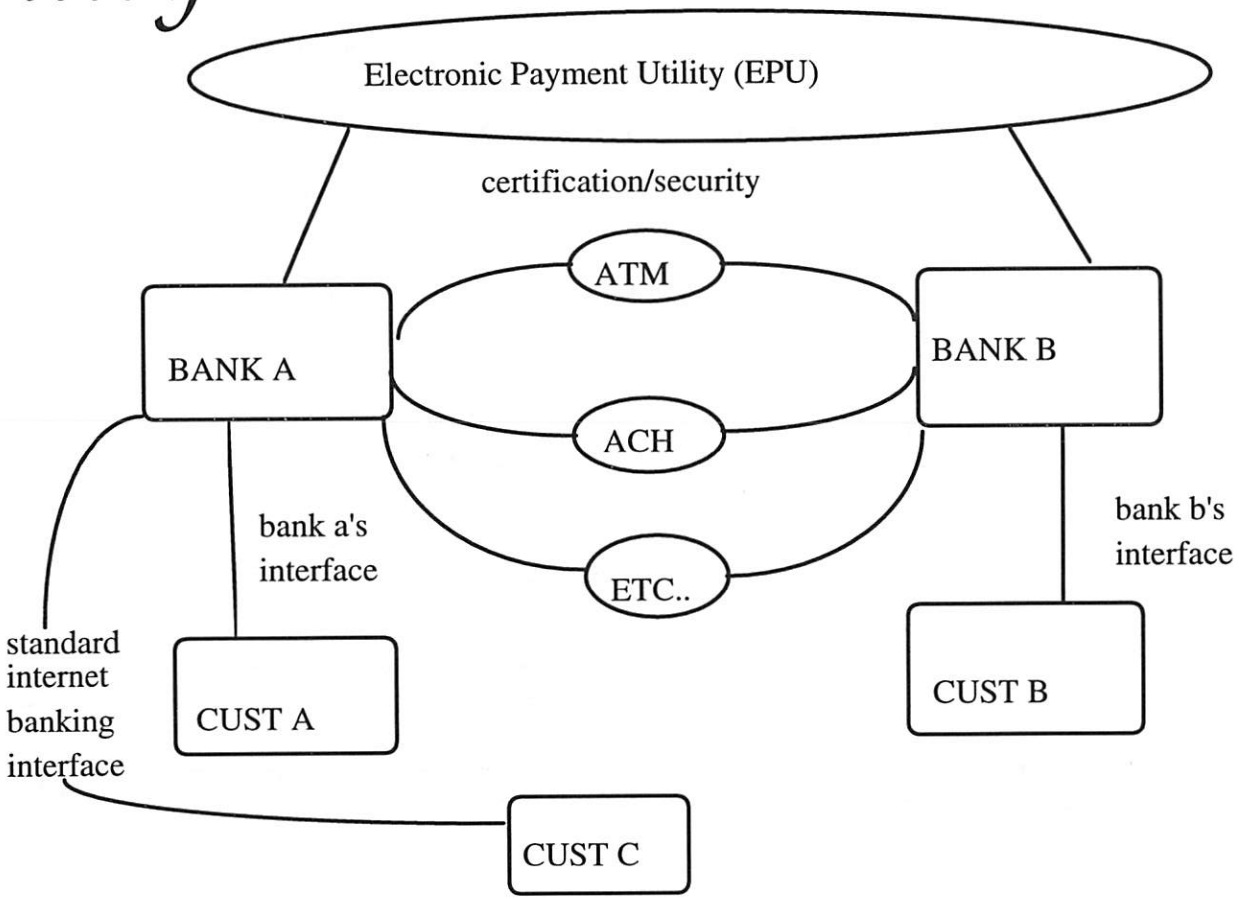
Desirable Design Objectives & Architecture

- Open & modular & widely accepted
- Able to initiate over a variety of mechanisms; e.g. POS devices, e-mail & eform enabled
- Interface to existing VAN's, payment and settlement/funds transfer systems
- Low cost of entry
- Does not require prior relationship
- Secure & private

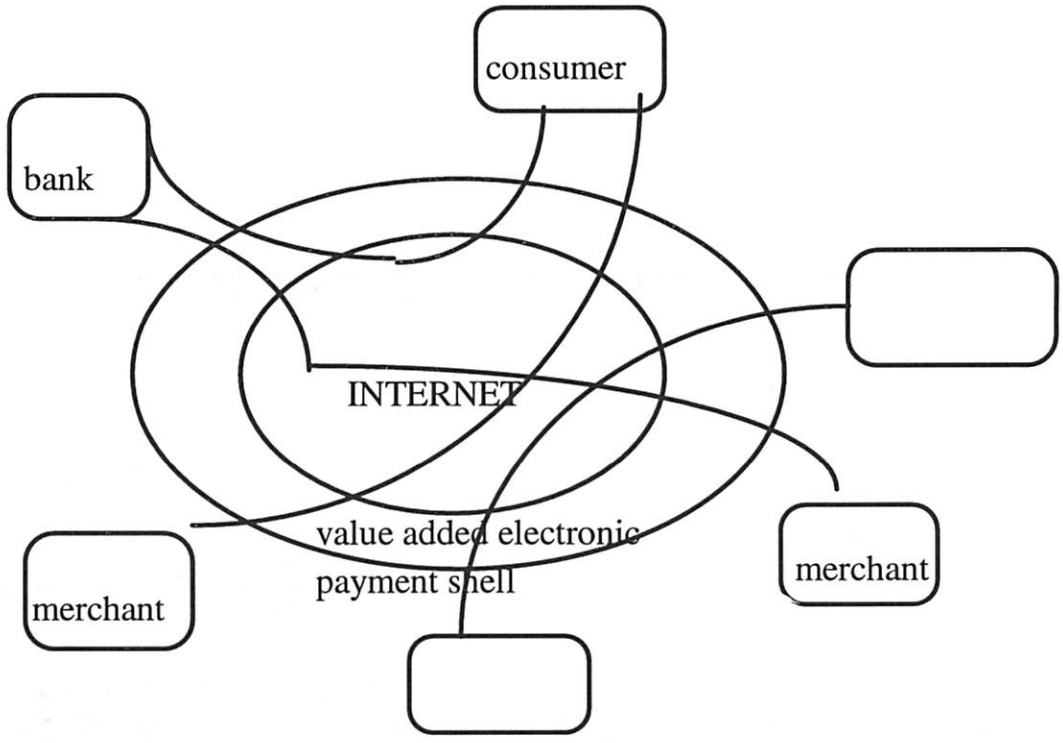
Electronic Commerce Goals

- On-line link between "net funds" and real funds & accounts
- High volume, low cost
- On-line Clearinghouse function
  - » **Timely**
  - » **Trusted (liquid, safe, final)**
  - » **Certain (reliable, accurate, safe, authenticated)**
  - » **Secure (safe from fraud, private)**
  - » **Affordable**
  - » **Accessible**
  - » **Scalable**

Interbank Electronic Payment Utility



Accessible from the Internet: A Secure Subnet Inside the Internet



Electronic Commerce Architectural Principles

- Open flexible architecture
- Minimum set of API's sufficient to accommodate all variations:
  - » **on-line single financial service firm**
  - » **on-line multiple financial service firm**
  - » **bilateral exchange - no on-line financial service firm**
- Assure interoperability amongst billing & payment schemes achieved

Electronic Commerce Outputs

- Functioning "Net" Clearinghouse established
- Identification and isolation of risks associated various payment schemes
- Minimum level of functionality & security assessed and certified
  - » **authenticated**
  - » **authorized**
  - » **"signed"**
  - » **private**

Electronic Check

- Payment mechanism for commerce over the network
- Can be extended to POS
- Based upon electronic analog to check, but can be extended to other payment types
    - **e.g. debit, credit, funds transfer**
- Allows two parties to transact on-line without need for third party
- Exchange via public communications (e.g. email, WWW) and a wide variety of devices

Electronic Check

- Open API
- Can attach remittance information
- Use of digital signatures for authentication
- One-way secure hash function/data integrity
- Encryption for privacy optional
- Eventual expansion for credit card, e-cash..
- Interfaces to existing banking payment & settlement networks; e.g. ECP & ACH

Interbank Check Imaging

- **1994 checks 80% non-cash payments**
- **Physical processing technologies have reached a plateau**
- **Current technologies attack less than 30% of costs**
- **Emerging imaging technologies present opportunities**
- **Greatest gain from eliminating exchange of paper check**
- **Reduce fraud opportunities**
  - **$ 5-10 billion/year, bad checks - retail**
  - **$ 800 million/year, bad checks - banks**

Interbank Check Imaging

Vision

- Payee or the bank of first deposit captures check image
- Physical movement of check stops at this point
- Check image integrated with check clearing system
- Customers have option to receive account statements with truncated check data or with check image
- Paper Checks-to-Digits & Digits-to-Paper

Fraud Detection & Management

- Commercial & Consumer
  - **Debit & Credit Cards initial focus**
- Orientation to public networks
- Physical, logical & process security
- Prevention, detection, damage control & post analysis

Fraud Project: A Phased Approach

Short Term

- Non Customer related data information share
- Fraud Detection Model (Neural Net, Advanced Stat Technique) evaluation testbed
- Biometric evaluation testbed

Long Term Work

- Model design / selection
- Temper resistant & Temper evident technology
- Vulnerability analysis & testbed

THE USENIX ASSOCIATION

USENIX is the UNIX and Advanced Computing Systems Technical and Professional Association. Since 1975, the USENIX Association has brought together the community of engineers, scientists, and technicians working on the cutting edge of the computing world. The USENIX conferences and technical symposia have become the essential meeting grounds for the presentation and discussion of the most advanced information on the developments of all aspects of computing systems.

USENIX and its members are dedicated to:

- problem-solving with a practical bias,
- fostering innovation and research that works,
- communicating rapidly the results of both research and innovation,
- providing a neutral forum for the exercise of critical thought and the airing of technical issues.

USENIX holds an annual technical conference, an annual system administration conference co-sponsored with SAGE, and single topic symposia throughout the year. It publishes *;login:*, a bi-monthly newsletter; *Computing Systems*, a quarterly technical journal published in association with The MIT Press; and conference proceedings for each of its conferences and symposia. It also sponsors local and special technical groups relevant to the UNIX environment as well as participating in various standards efforts.

SAGE, the System Administrators Guild

The System Administrators Guild, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX. SAGE activities include the publishing of *Job Descriptions for System Administrators*, edited by Tina Darmohray; "SAGE News", a regular feature in *;login:*, The System Administrator Profile, an annual survey of system administrator salaries and responsibilities; support of working groups; and an archive site for papers from the System Administration Conferences.

Member Benefits:

- Free subscription to *;login:*, the Association's bi-monthly newsletter featuring technical articles, a worldwide calendar of events, SAGE News, media reviews, summaries of conferences, Snitch Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts, and much more.
- Free subscription to *Computing Systems*, a refereed technical quarterly published with The MIT Press.
- Discounts on registration for technical sessions at all USENIX conferences and symposia.
- Discounts on proceedings from USENIX conferences and symposia.
- Access to the Online Library of all 1993-1995 published USENIX proceedings on the USENIX World Wide Web site
- Discount on the new 4.4BSD Manuals, the definitive release of the Berkeley version of UNIX with a CD-ROM published by the USENIX Association and O'Reilly & Associates, Inc.
- Savings on *The Evolution of C++: Language Design in the Marketplace of Ideas*, edited by Jim Waldo of Sun Microsystems Laboratories, the USENIX Association book published by The MIT Press.
- Special subscription rates to *UniForum Monthly*, *UniNews* and the annual *UniForum Open Systems Products Directory* and *Linux Journal*
- Savings on selected titles from McGraw-Hill, The MIT Press, Prentice Hall, John Wiley & Sons, O'Reilly and Associates, and UniForum.

Supporting Members of the USENIX Association:

ANDATACO
Apunix Computer Services
Frame Technology Corporation
GraphOn Corporation
Matsushita Electrical Industrial Co., Ltd.
Shiva Corporation
Sun Microsystems, Inc., Sunsoft Network Products
Tandem Computers, Inc.
UUNET Technologies, Inc

SAGE Supporting Members: Enterprise Systems Management Corporation and Great Circle Associates

For further information about membership, conferences or publications, contact:

The USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 USA
Email: *office@usenix.org*
Phone: +1-510-528-8649
Fax: +1-510-548-5738
WWW URL: *http://www.usenix.org*

ISBN 1-880446-74-X